Synchronizing Automata and the Road Coloring Theorem

M. V. Volkov

Ural State University, Ekaterinburg, Russia

Synchronizing automata

We consider DFA: $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

Synchronizing automata

We consider DFA: $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

$\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

Synchronizing automata

We consider DFA: $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

$\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$. $|Q\,.\,w| = 1$. Here $Q\,.\,v = \{\delta(q, v) \mid q \in Q\}$.

We consider DFA: $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

$\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$. $|Q \,.\, w| = 1$. Here $Q \,.\, v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a *reset word* for $\mathscr{A}$.

Synchronizing automata

We consider DFA: $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

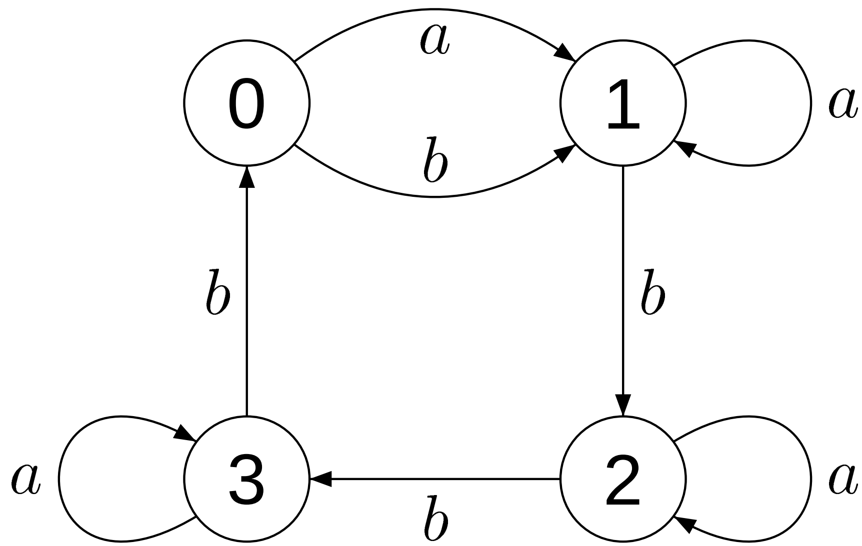
$\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$. $|Q \,.\, w| = 1$. Here $Q \,.\, v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a *reset word* for $\mathscr{A}$.

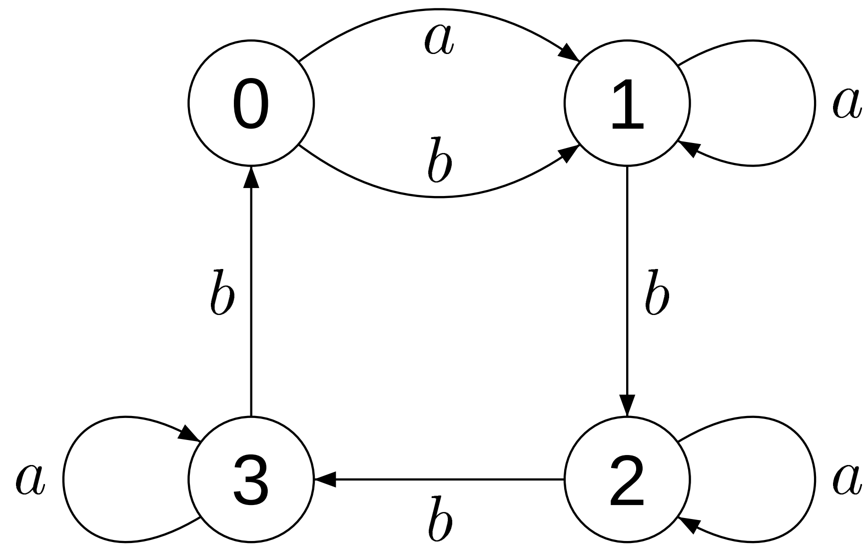
Other names:

- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

Synchronizing automata



Synchronizing automata



A reset word is $abbbabbba$. Applying it at any state brings the automaton to the state 1.

Synchronizing Automata

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14, no.3, 208–216 [in Slovak]) though implicitly it had been around since at least 1956 (see proceedings).

Synchronizing Automata

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14, no.3, 208–216 [in Slovak]) though implicitly it had been around since at least 1956 (see proceedings).

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Synchronizing Automata

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14, no.3, 208–216 [in Slovak]) though implicitly it had been around since at least 1956 (see proceedings).

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while “behind” the Moon (Černý’s original motivation).

Synchronizing Automata

It is not surprising that synchronizing automata were re-invented a number of times:

Synchronizing Automata

It is not surprising that synchronizing automata were re-invented a number of times:

- The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the 1960s.

Synchronizing Automata

It is not surprising that synchronizing automata were re-invented a number of times:

- The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the 1960s.
- Černý's paper published in Slovak language remained unknown in the English-speaking world for quite a long time.

Synchronizing Automata

It is not surprising that synchronizing automata were re-invented a number of times:

- The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the 1960s.
- Černý's paper published in Slovak language remained unknown in the English-speaking world for quite a long time.

Example: A. E. Laemmel, B. Rudner, Study of the application of coding theory, Report PIBEP-69-034, Polytechnic Inst. Brooklyn, Dept. Electrophysics, Farmingdale, N.Y., 94 pp.; more examples in proceedings.

Re-inventing by Engineers

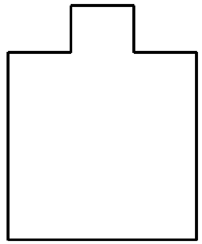
Since the 60s and till the 90s synchronizing automata were considered as a useful tool for testing of reactive systems (first circuits, later protocols).

Re-inventing by Engineers

Since the 60s and till the 90s synchronizing automata were considered as a useful tool for testing of reactive systems (first circuits, later protocols).
In the 80s, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.

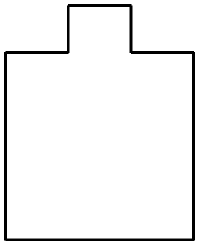
Re-inventing by Engineers

Since the 60s and till the 90s synchronizing automata were considered as a useful tool for testing of reactive systems (first circuits, later protocols).
In the 80s, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.
Suppose that one of the parts of a certain device has the following shape:



Re-inventing by Engineers

Since the 60s and till the 90s synchronizing automata were considered as a useful tool for testing of reactive systems (first circuits, later protocols).
In the 80s, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.
Suppose that one of the parts of a certain device has the following shape:



Such parts arrive at manufacturing sites in boxes and they need to be sorted and oriented before assembly.

Re-inventing by Engineers

Assume that only four initial orientations of the part shown above are possible, namely, the following ones:

Re-inventing by Engineers

Assume that only four initial orientations of the part shown above are possible, namely, the following ones:

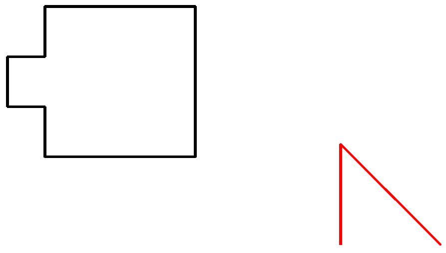
Suppose that prior the assembly the part should take the “bump-left” orientation (the second one in the picture). Thus, one has to construct an orienter which action will put the part in the prescribed position independently of its initial orientation.

Re-inventing by Engineers

We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.

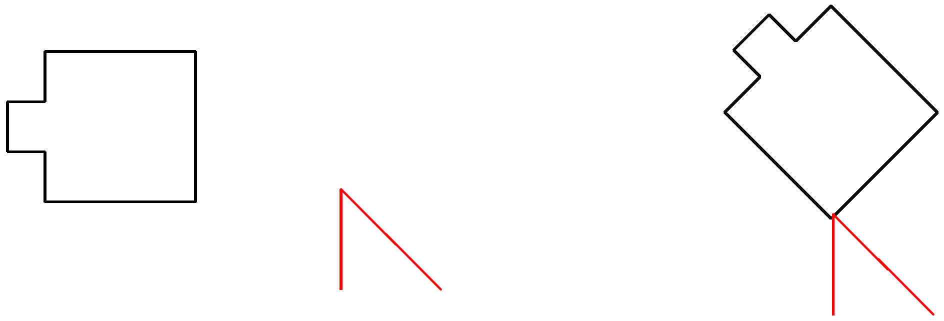
Re-inventing by Engineers

We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.
A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.



Re-inventing by Engineers

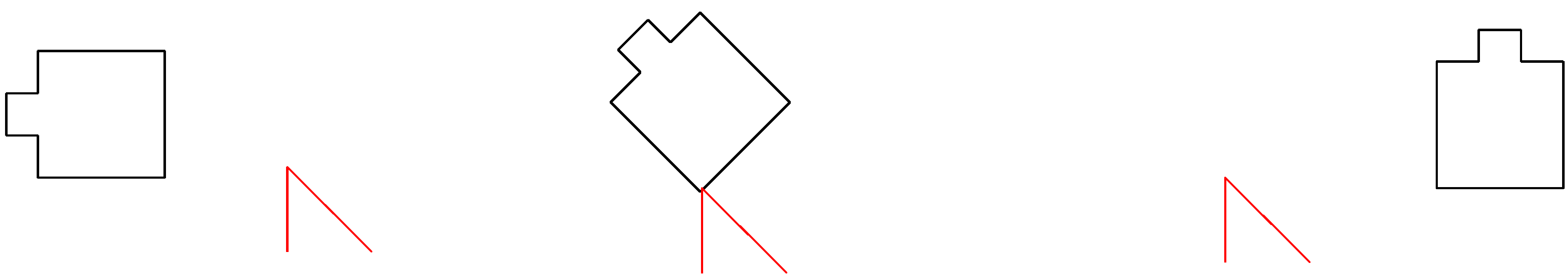
We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.
A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.



Being curried by the belt, the part then is forced to turn 90° clockwise.

Re-inventing by Engineers

We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.
A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.



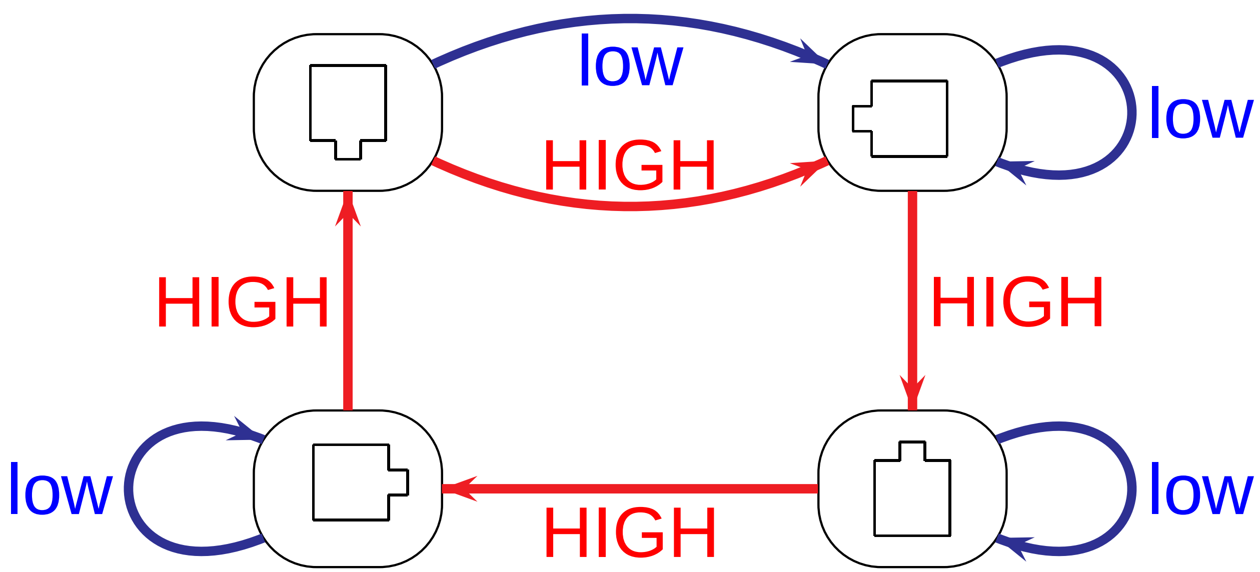
Being curried by the belt, the part then is forced to turn 90° clockwise.

Re-inventing by Engineers

A low obstacle has the same effect whenever the part is in the “bump-down” orientation; otherwise it does not touch the part which therefore passes by without changing the orientation.

Re-inventing by Engineers

A low obstacle has the same effect whenever the part is in the “bump-down” orientation; otherwise it does not touch the part which therefore passes by without changing the orientation.
The following schema summarizes how the obstacles effect the orientation of the part in question:

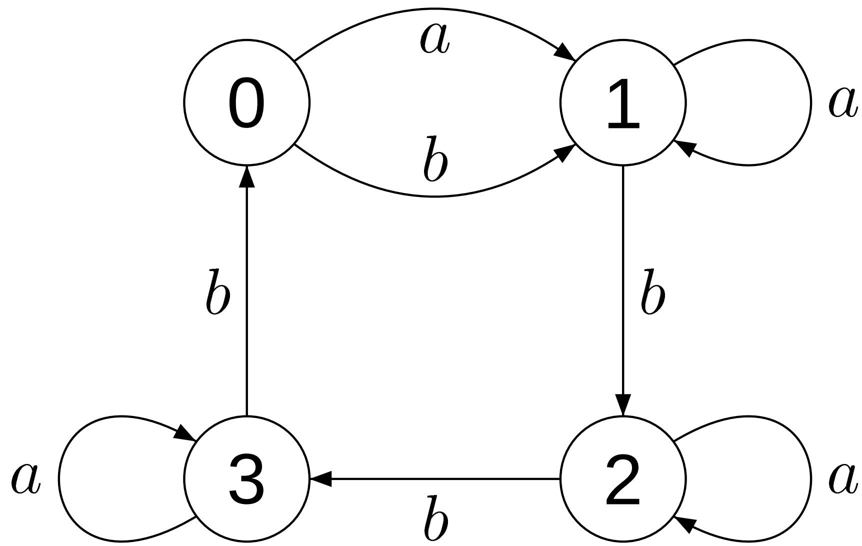


Re-inventing by Engineers

We met this picture a few slides ago:

Re-inventing by Engineers

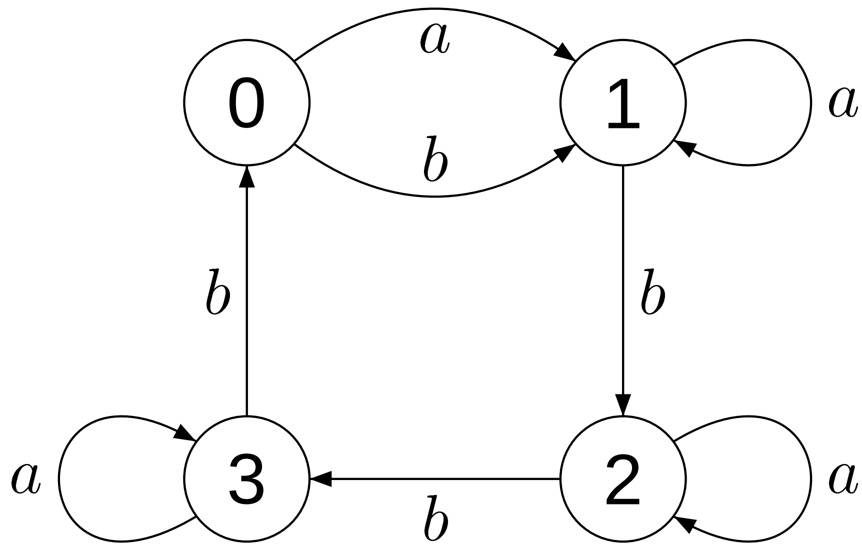
We met this picture a few slides ago:



– this was our example of a synchronizing automaton, and we saw that $abbbabbba$ is a reset sequence of actions.

Re-inventing by Engineers

We met this picture a few slides ago:



– this was our example of a synchronizing automaton, and we saw that $abbbabbba$ is a reset sequence of actions. Hence the series of obstacles

low-HIGH-HIGH-HIGH-low-HIGH-HIGH-HIGH-low

yields the desired sensorless orienter.

Possible Use in Biocomputing

In DNA-computing, there is a fast progressing work by Ehud Shapiro’s group on “*soup of automata*” (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc).

Possible Use in Biocomputing

In DNA-computing, there is a fast progressing work by Ehud Shapiro's group on "*soup of automata*" (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc). They have produced a solution containing 3×10^{12} identical DNA-based automata per μl.

Possible Use in Biocomputing

In DNA-computing, there is a fast progressing work by Ehud Shapiro's group on "*soup of automata*" (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc). They have produced a solution containing 3×10^{12} identical DNA-based automata per μl. These automata can work in parallel on different inputs (DNA strands), thus ending up in different and unpredictable states.

Possible Use in Biocomputing

In DNA-computing, there is a fast progressing work by Ehud Shapiro's group on "*soup of automata*" (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc). They have produced a solution containing 3×10^{12} identical DNA-based automata per μl. These automata can work in parallel on different inputs (DNA strands), thus ending up in different and unpredictable states. One has to feed the automata with an reset sequence (again encoded by a DNA-strand) in order to get them ready for a new use.

Outline of the Tutorial

- From the viewpoint of applications, real or yet imaginary, algorithmic issues are of crucial importance. We discuss them in Part I.

Outline of the Tutorial

- From the viewpoint of applications, real or yet imaginary, algorithmic issues are of crucial importance. We discuss them in Part I.
- Synchronizing automata constitute an interesting combinatorial object. Their studies are mainly motivated by the Černý conjecture. We discuss the Černý conjecture in Part II.

Outline of the Tutorial

- From the viewpoint of applications, real or yet imaginary, algorithmic issues are of crucial importance. We discuss them in Part I.
- Synchronizing automata constitute an interesting combinatorial object. Their studies are mainly motivated by the Černý conjecture. We discuss the Černý conjecture in Part II.
- Yet another mathematical motivation for studying synchronizing automata comes from symbolic dynamics. In Part III we present a recent breakthrough in the area—a (positive) solution to the Road Coloring Problem found by Trahtman.

Checking Synchronizability

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?*

Checking Synchronizability

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic power automaton construction.

Checking Synchronizability

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic power automaton construction.

The *power automaton* $\mathcal{P}(\mathscr{A})$ of a given DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$:
- states are the non-empty subsets of Q,
- $\delta(P, a) = P \,.\, a = \{\delta(p, a) \mid p \in P\}$

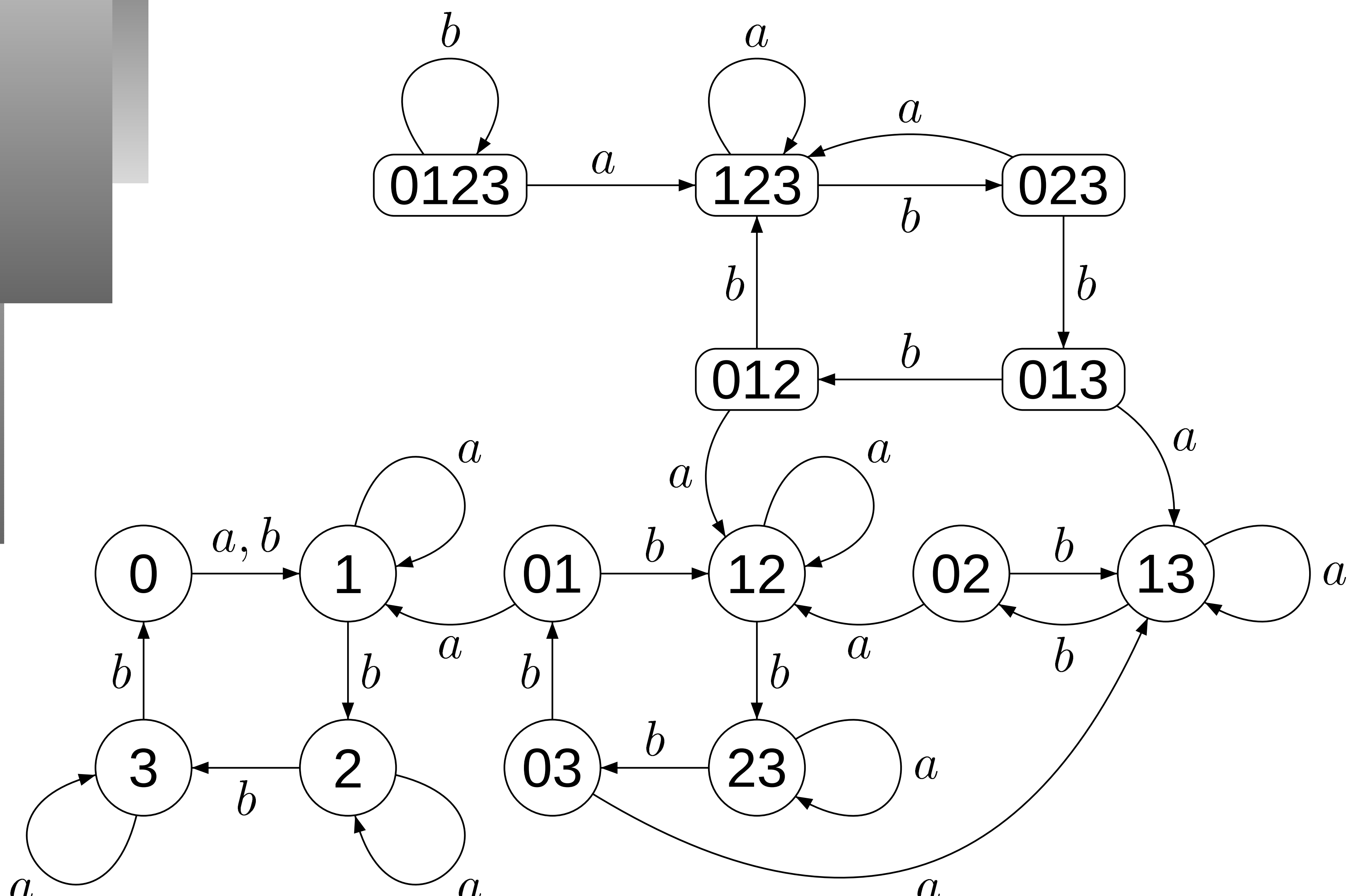
Checking Synchronizability

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic power automaton construction.

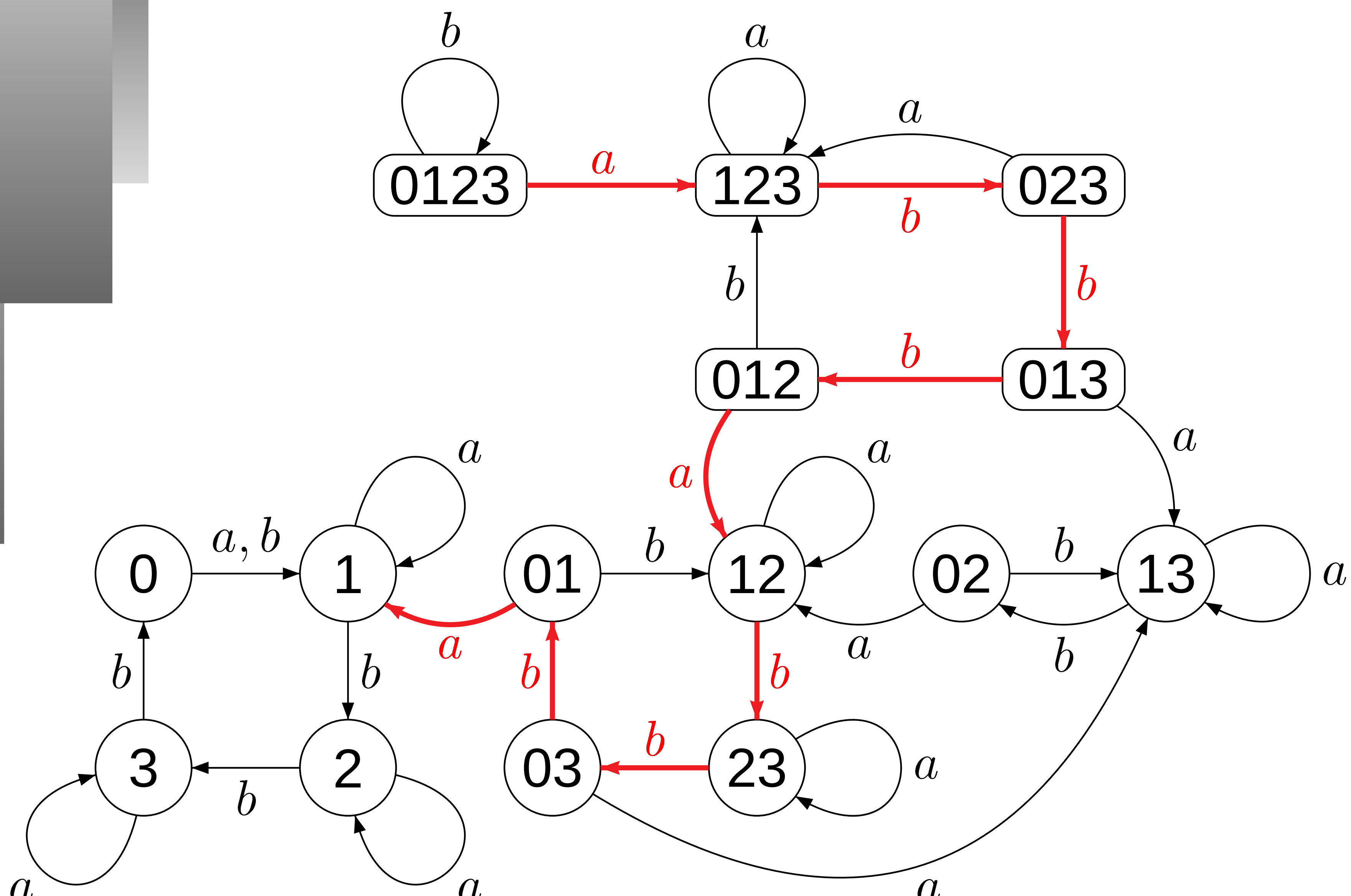
The *power automaton* $\mathcal{P}(\mathscr{A})$ of a given DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$:
- states are the non-empty subsets of Q,
- $\delta(P, a) = P \,.\, a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA $\mathscr{A}$ iff w labels a path in $\mathcal{P}(\mathscr{A})$ starting at Q and ending at a singleton.

Checking Synchronizability



Checking Synchronizability



Checking Synchronizability

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS.

Checking Synchronizability

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of $\mathscr{A}$.

Checking Synchronizability

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of $\mathscr{A}$.

The following result by Černý gives a polynomial algorithm:

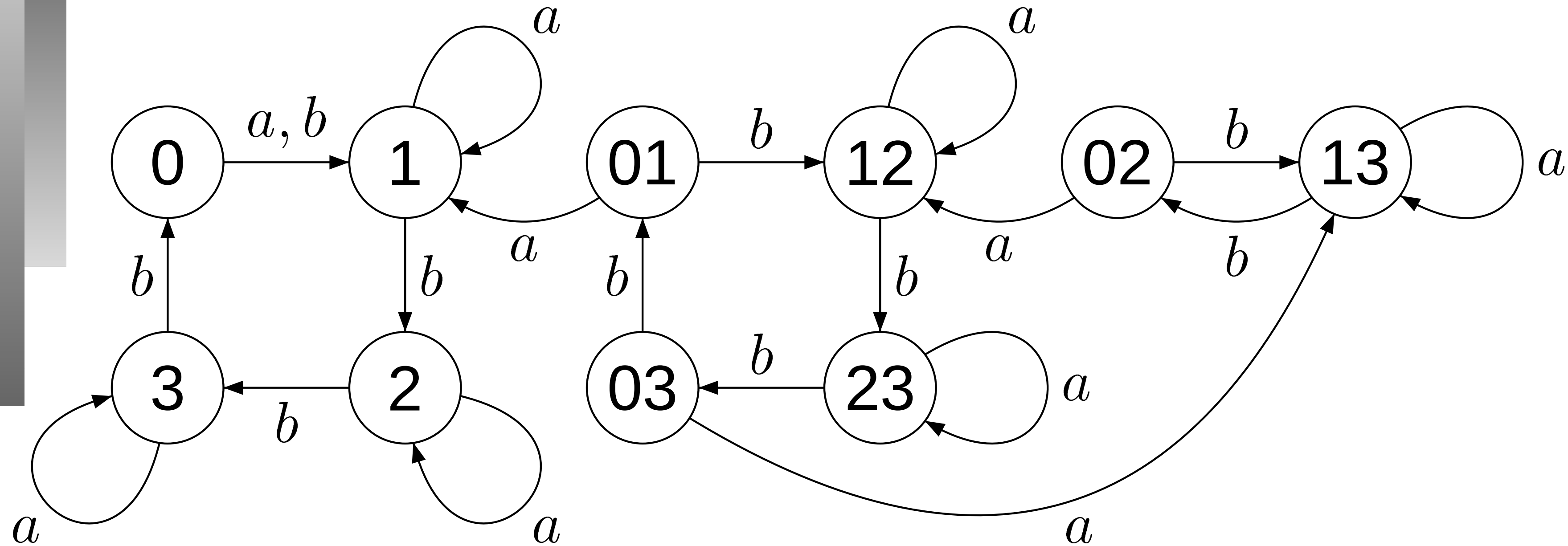
Checking Synchronizability

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of $\mathscr{A}$.

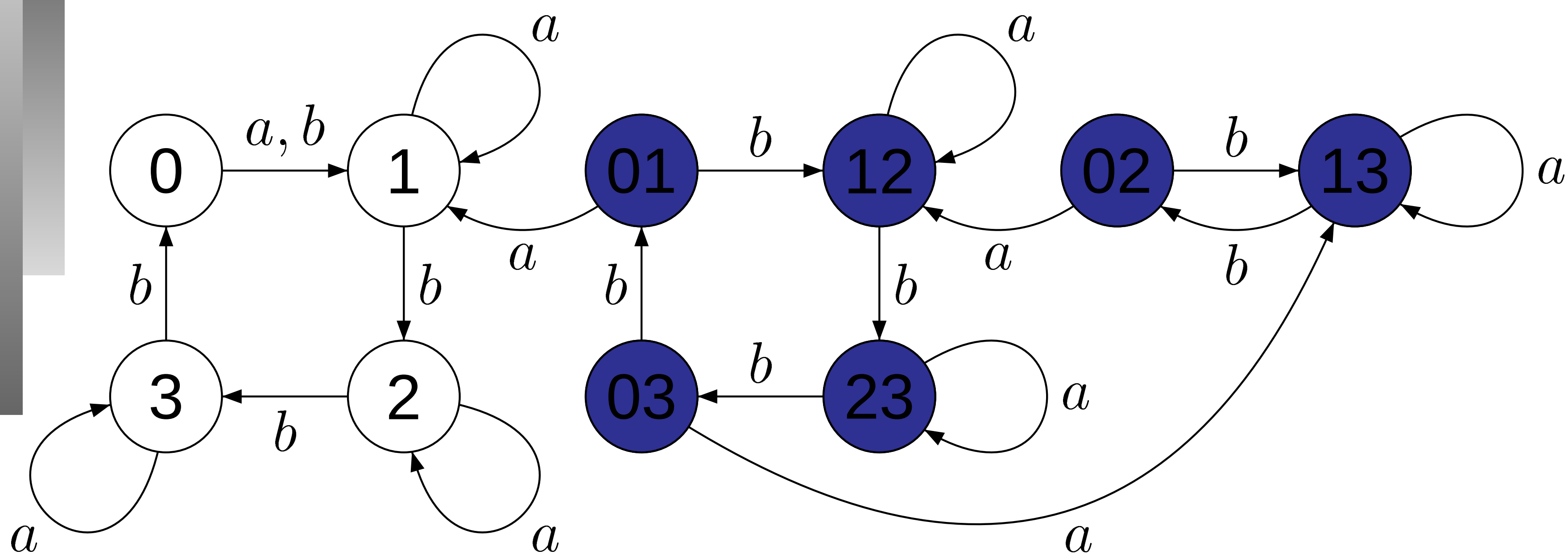
The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA* $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ *is synchronizing iff for every* $q, q' \in Q$ *there exists a word* $w \in \Sigma^*$ *such that* $\delta(q, w) = \delta(q', w)$.

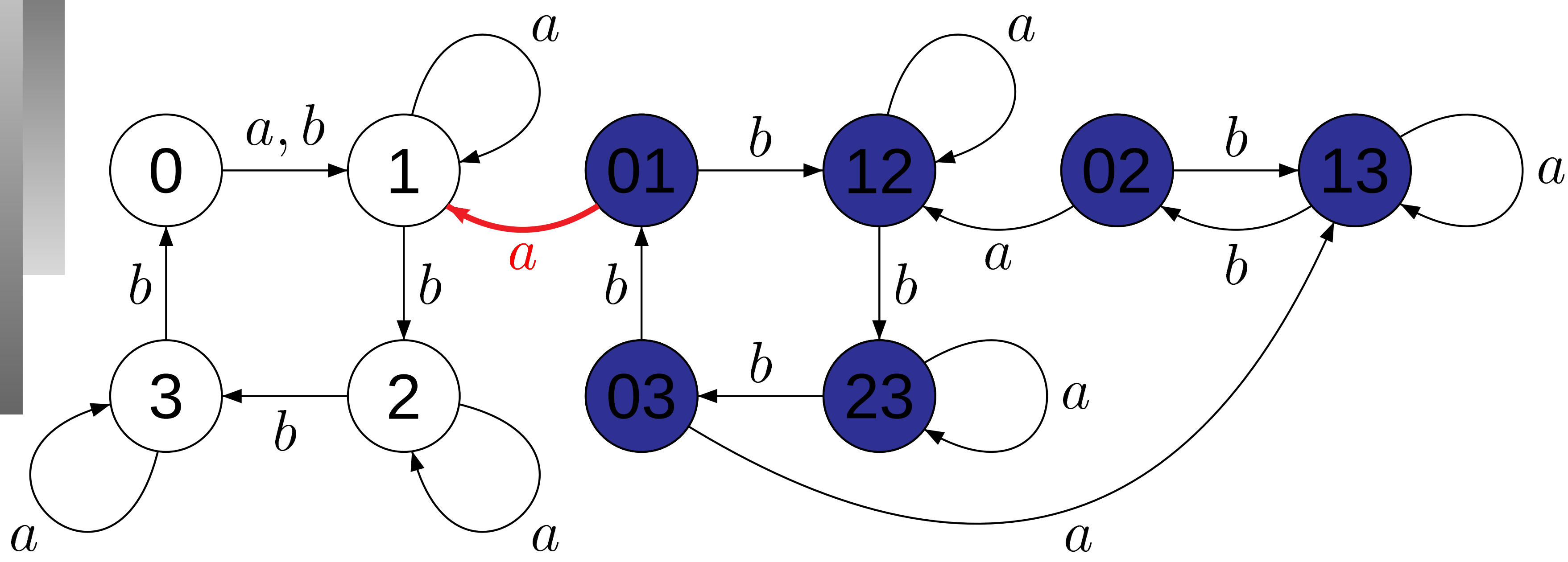
Checking Synchronizability



Checking Synchronizability

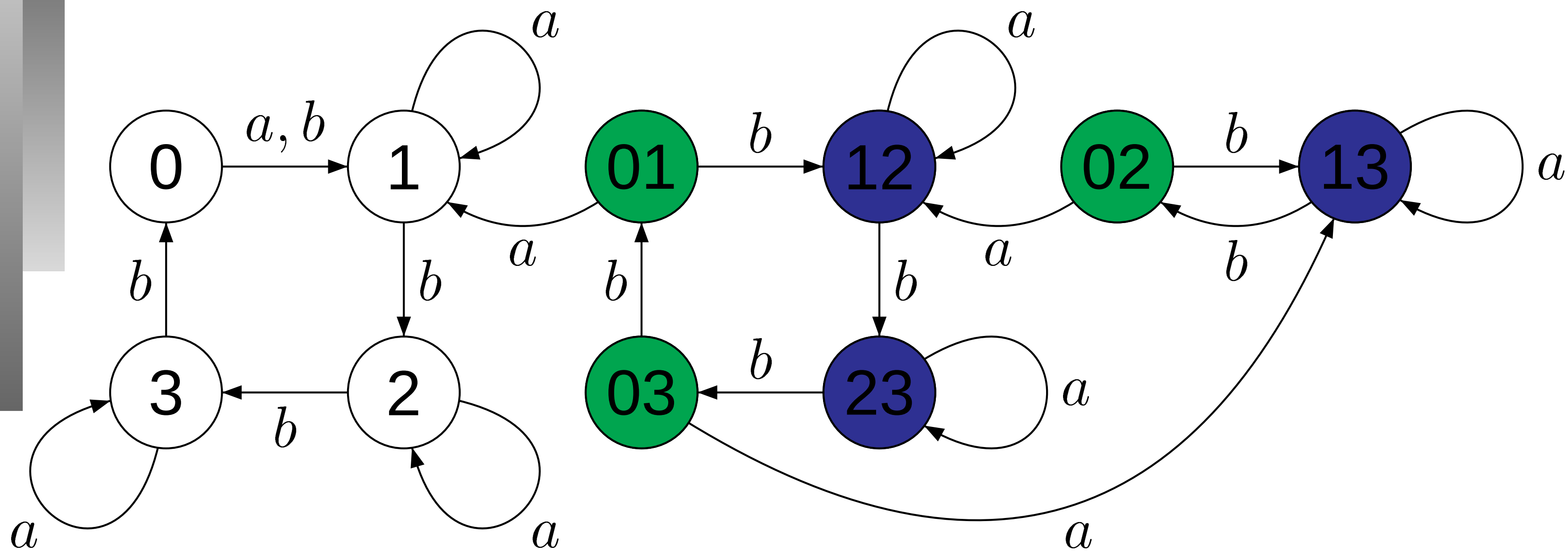


Checking Synchronizability

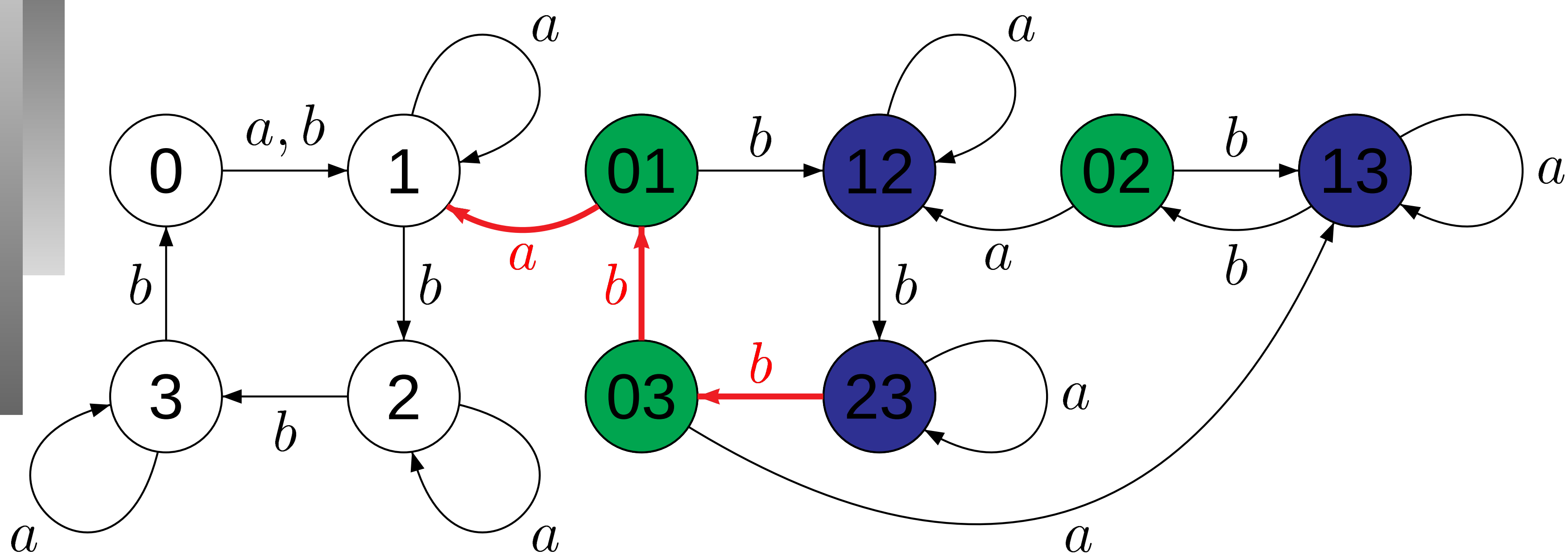


$$a,\ Q\,.\,a = \{1, 2, 3\}$$

Checking Synchronizability

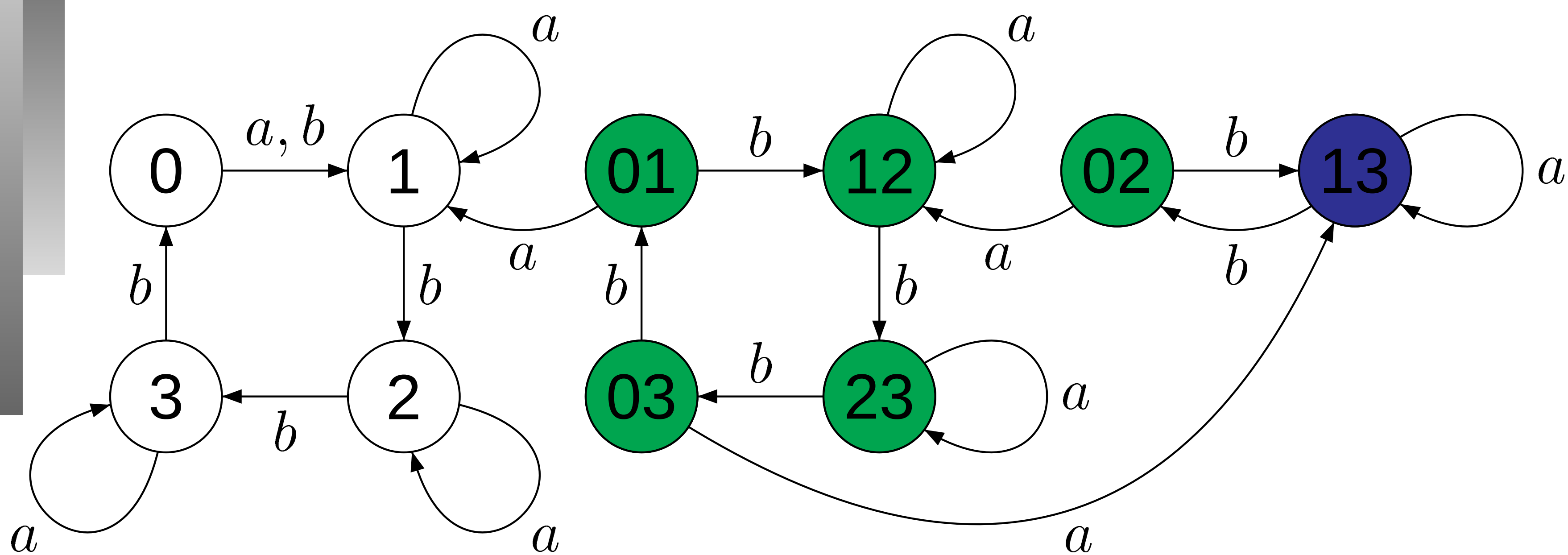


Checking Synchronizability

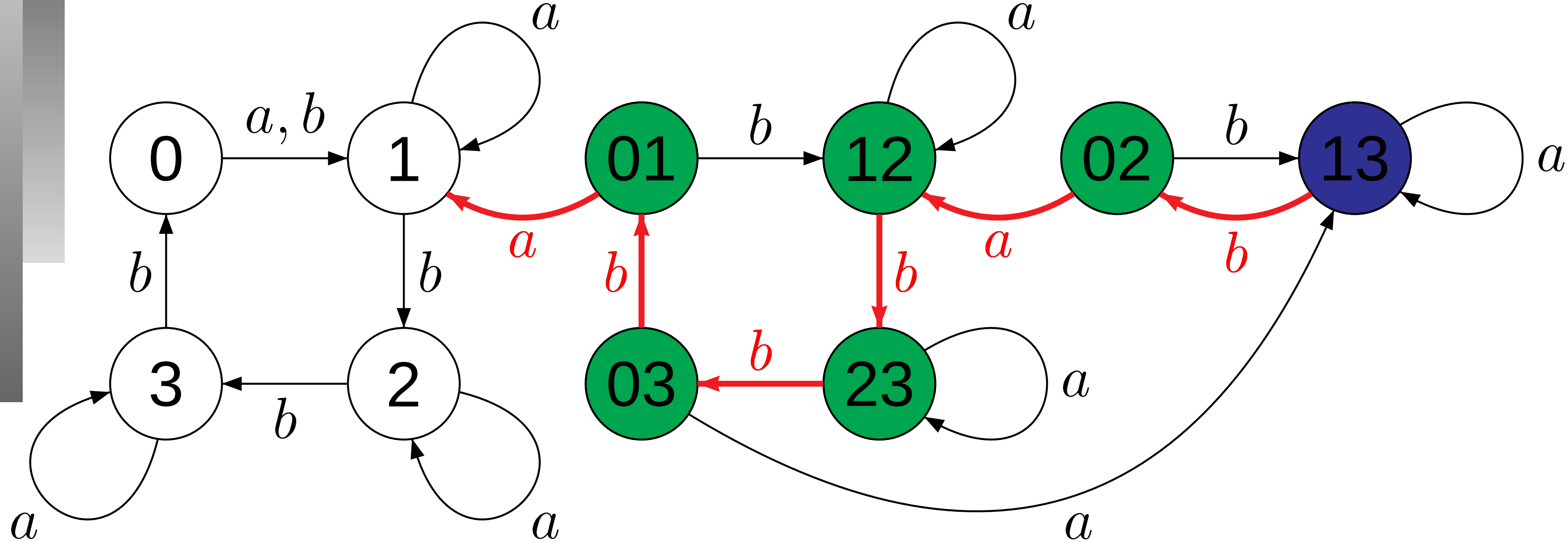


$$a \cdot bba,\ \ Q \,.\, abba = \{1, 3\}$$

Checking Synchronizability

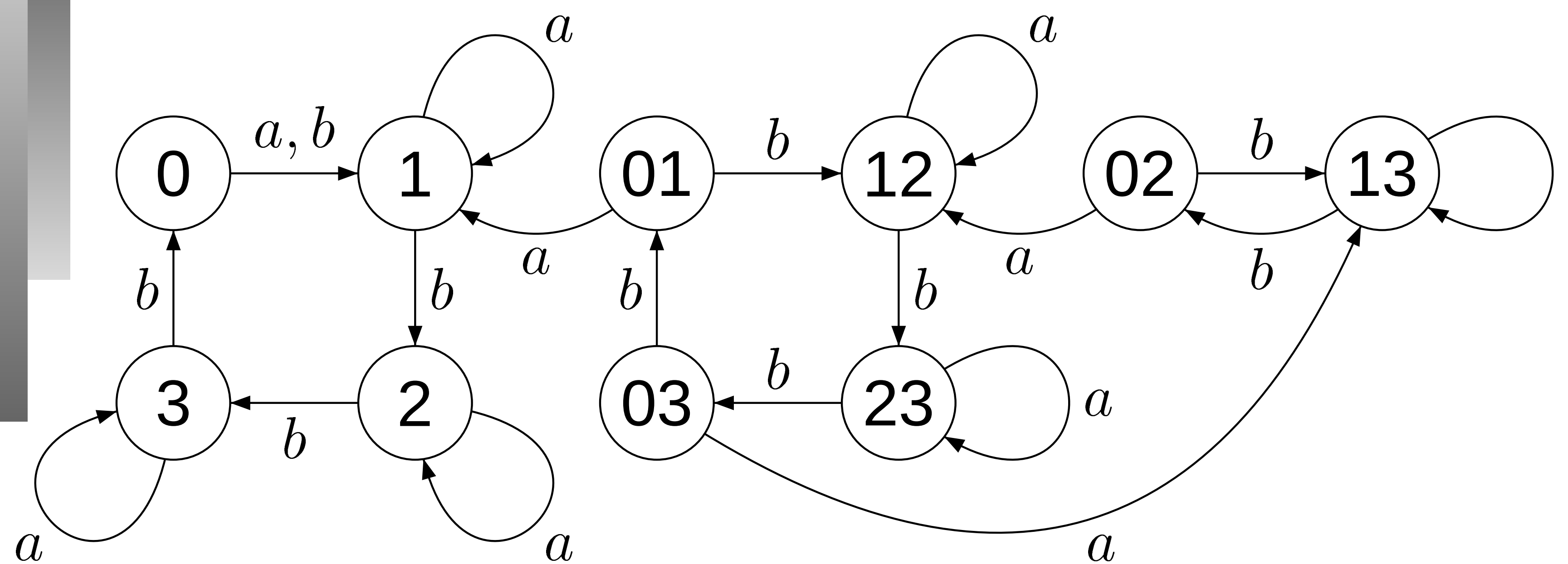


Checking Synchronizability



$$abba \cdot babbba, \ Q \,.\, abbababbba = \{1\}$$

Checking Synchronizability



$$abba \cdot babbba,\ Q\,.\,abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9 for this automaton.

Checking Synchronizability

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q.

Checking Synchronizability

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

Checking Synchronizability

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$. If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Checking Synchronizability

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$. If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$:

Checking Synchronizability

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$. If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n-1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of green 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$.

Checking Synchronizability

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$. If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time.

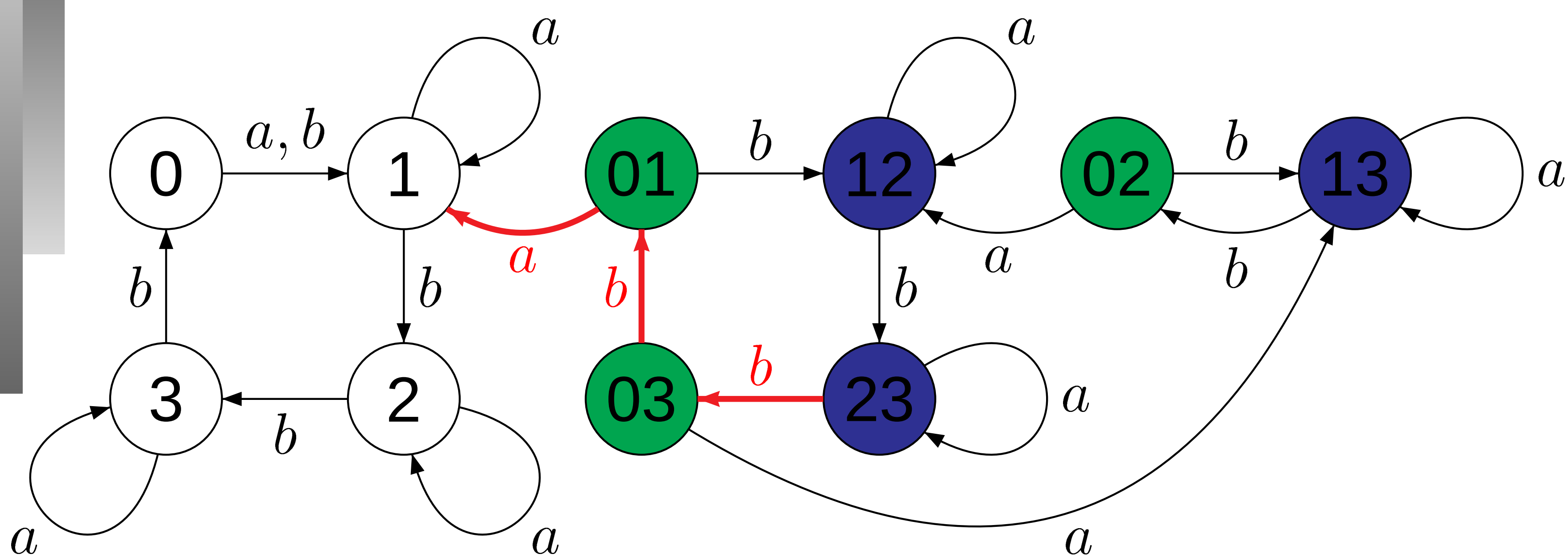
Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of green 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3-n}{3}$.

Checking Synchronizability

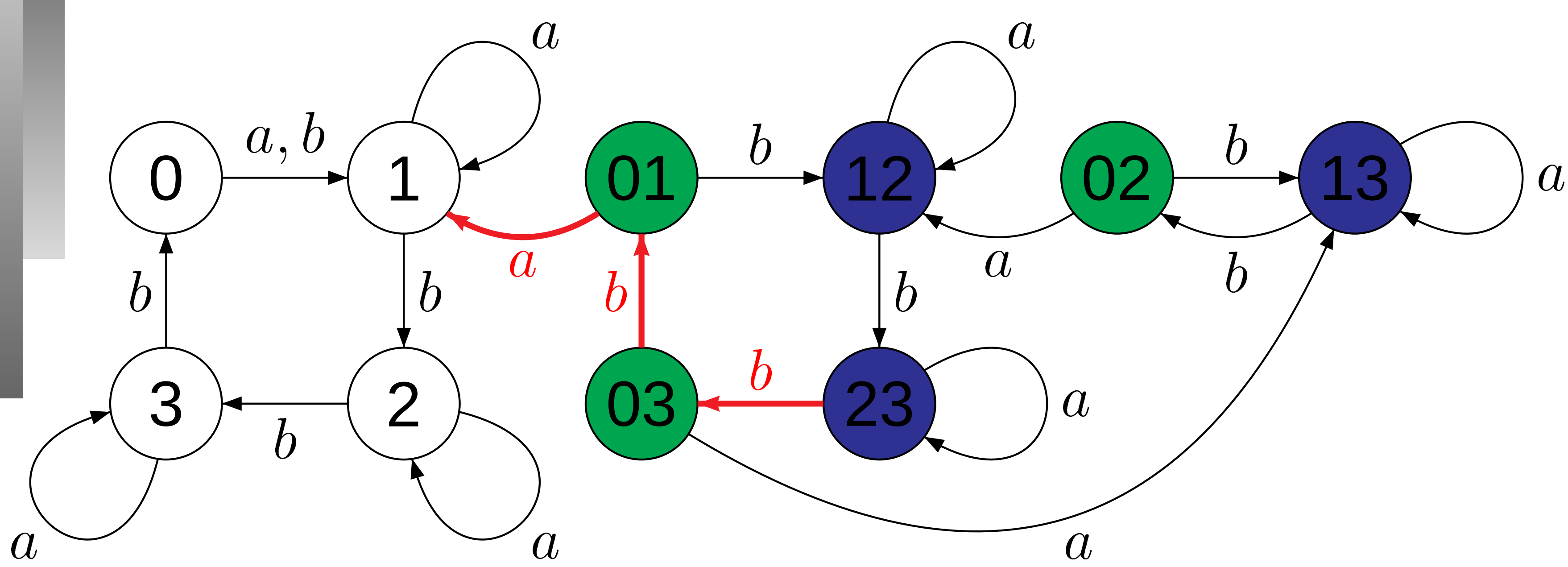
Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$. If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n-1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of green 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3-n}{3}$. Can we do better? What is the exact bound?

Studying Greedy Algorithm

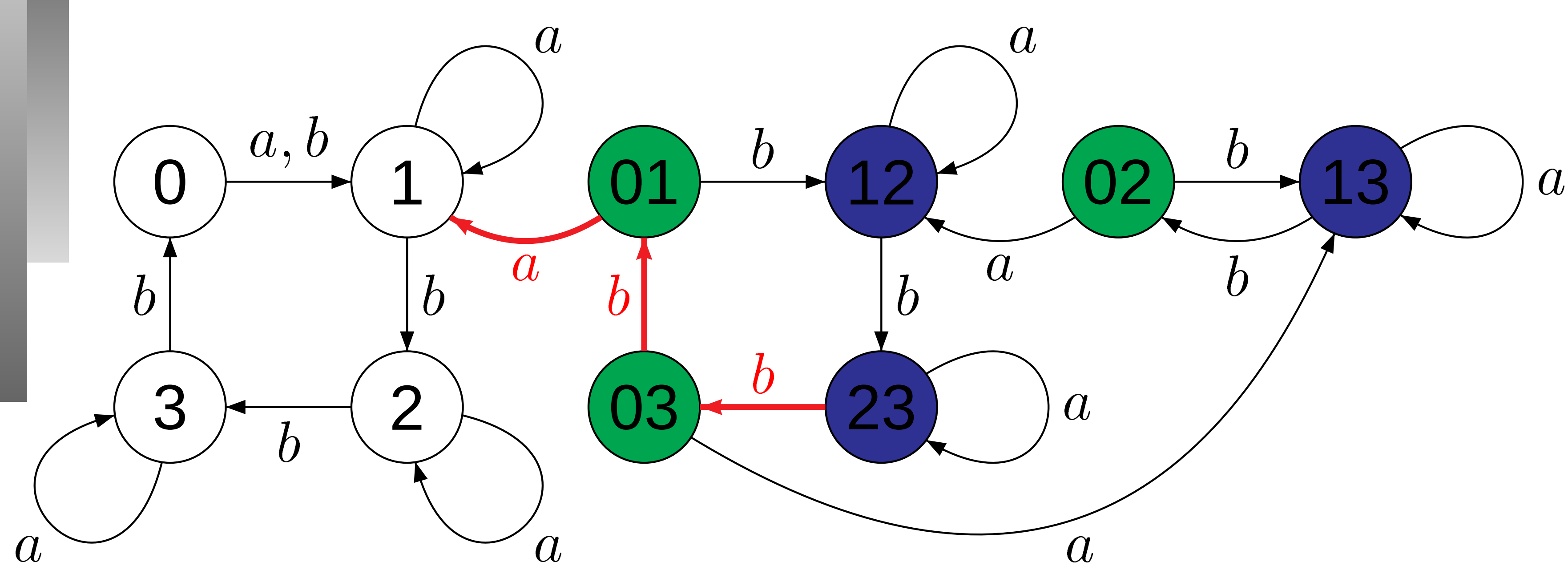


Studying Greedy Algorithm



We see that the shortest path from a blue 2-subset to a singleton do not necessarily pass through all green 2-subsets.

Studying Greedy Algorithm



We see that the shortest path from a blue 2-subset to a singleton do not necessarily pass through all green 2-subsets.
Consider a generic step of the algorithm at which states to be compressed form a set P with $|P| = k > 1$ and let $v = a_1 \cdots a_\ell$ with $a_i \in \Sigma$, $i = 1, \ldots, \ell$, be a word of minimum length such that $|P \,.\, v| < k$.

Studying Greedy Algorithm

The sets $P_1 = P,\ P_2 = P_1 \,.\, a_1,\ \ldots,\ P_\ell = P_{\ell-1} \,.\, a_{\ell-1}$ are k-subsets of Q.

Studying Greedy Algorithm

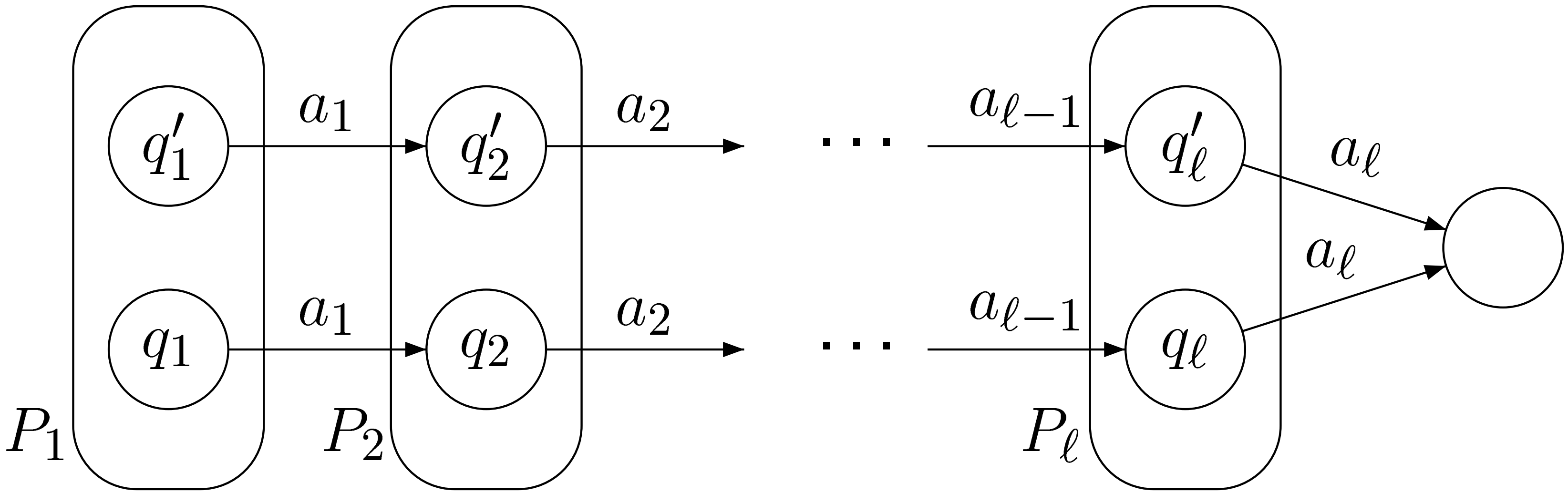
The sets $P_1 = P,\ P_2 = P_1 \, . \, a_1,\ \ldots,\ P_\ell = P_{\ell-1} \, . \, a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell \, . \, a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$.

Studying Greedy Algorithm

The sets $P_1 = P,\ P_2 = P_1 \,.\, a_1,\ \ldots,\ P_\ell = P_{\ell-1} \,.\, a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell \,.\, a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$. Now define 2-subsets $R_i = \{q_i, q'_i\} \subseteq P_i$, $i = 1, \ldots, \ell$, such that $\delta(q_i, a_i) = q_{i+1}$, $\delta(q'_i, a_i) = q'_{i+1}$ for $i = 1, \ldots, \ell - 1$.

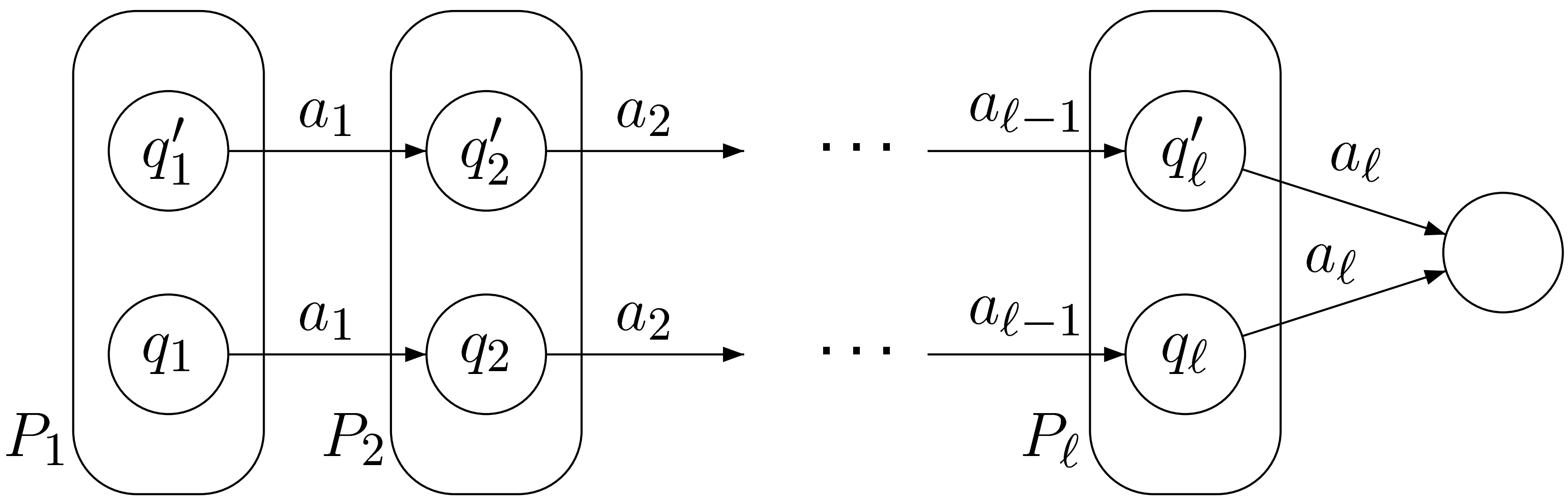
Studying Greedy Algorithm

The sets $P_1 = P,\ P_2 = P_1 \,.\, a_1,\ \ldots,\ P_\ell = P_{\ell-1} \,.\, a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell \,.\, a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$. Now define 2-subsets $R_i = \{q_i, q'_i\} \subseteq P_i$, $i = 1, \ldots, \ell$, such that $\delta(q_i, a_i) = q_{i+1}$, $\delta(q'_i, a_i) = q'_{i+1}$ for $i = 1, \ldots, \ell - 1$.



Studying Greedy Algorithm

The sets $P_1 = P,\ P_2 = P_1 \,.\, a_1,\ \ldots,\ P_\ell = P_{\ell-1} \,.\, a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell \,.\, a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$. Now define 2-subsets $R_i = \{q_i, q'_i\} \subseteq P_i$, $i = 1, \ldots, \ell$, such that $\delta(q_i, a_i) = q_{i+1}$, $\delta(q'_i, a_i) = q'_{i+1}$ for $i = 1, \ldots, \ell - 1$.



The condition that v is a word of minimum length with $|P \,.\, v| < |P|$ implies $R_i \not\subseteq P_j$ for $1 \leq j < i \leq \ell$.

Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \ldots, P_\ell$ a sequence of its k-subsets $(k > 1)$ such that each P_i, $1 < i \leq \ell$, includes a “fresh” 2-subset that does not occur in any pervious P_j ($1 \leq j < i$).

Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \ldots, P_\ell$ a sequence of its k-subsets $(k > 1)$ such that each P_i, $1 < i \leq \ell$, includes a “fresh” 2-subset that does not occur in any pervious P_j $(1 \leq j < i)$. How long can such *refreshing* sequences be?

Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \ldots, P_\ell$ a sequence of its k-subsets $(k > 1)$ such that each P_i, $1 < i \leq \ell$, includes a “fresh” 2-subset that does not occur in any pervious P_j ($1 \leq j < i$). How long can such *refreshing* sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list.

Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \ldots, P_\ell$ a sequence of its k-subsets $(k > 1)$ such that each P_i, $1 < i \leq \ell$, includes a “fresh” 2-subset that does not occur in any pervious P_j ($1 \leq j < i$). How long can such *refreshing* sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the refreshing sequence $T_1, \ldots, T_s$ of length $s = \binom{n-k+2}{2}$.

Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \ldots, P_\ell$ a sequence of its k-subsets $(k > 1)$ such that each P_i, $1 < i \leq \ell$, includes a “fresh” 2-subset that does not occur in any pervious P_j ($1 \leq j < i$). How long can such *refreshing* sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the refreshing sequence $T_1, \ldots, T_s$ of length $s = \binom{n-k+2}{2}$. Is this the maximum?

Combinatorial Configuration

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127). See proceedings for a detailed history.

Combinatorial Configuration

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127). See proceedings for a detailed history.

The proof uses linearization techniques which is quite common in combinatorics of finite sets.

Combinatorial Configuration

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127). See proceedings for a detailed history.

The proof uses linearization techniques which is quite common in combinatorics of finite sets. One reformulates the problem in linear algebra terms and then uses the corresponding machinery.

Combinatorial Configuration

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127). See proceedings for a detailed history.

The proof uses linearization techniques which is quite common in combinatorics of finite sets. One reformulates the problem in linear algebra terms and then uses the corresponding machinery.

We identify Q with $\{1, 2, \ldots, n\}$ and assign to each k-subset $I = \{i_1, \ldots, i_k\}$ the following polynomial $D(I)$ in variables $x_{i_1}, \ldots, x_{i_k}$ over the field of rationals.

Combinatorial Configuration

$$I = \{i_1, \ldots, i_k\} \mapsto D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k \times k}$$

Combinatorial Configuration

$$I = \{i_1, \ldots, i_k\} \mapsto D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k \times k}$$

Then one proves that:
• the polynomials $D(P_1), \ldots, D(P_\ell)$ are linearly independent whenever the k-subsets $P_1, \ldots, P_\ell$ form a refreshing sequence;

Combinatorial Configuration

$$I = \{i_1, \ldots, i_k\} \mapsto D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k\times k}$$

Then one proves that:
• the polynomials $D(P_1), \ldots, D(P_\ell)$ are linearly independent whenever the k-subsets $P_1, \ldots, P_\ell$ form a refreshing sequence;
• the polynomials $D(T_1), \ldots, D(T_s)$ (derived from the "standard" sequence) generate the linear space spanned by all polynomials of the form $D(I)$.

Studying Greedy Algorithm

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$.

Studying Greedy Algorithm

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Summing up over $k = n, \ldots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

Studying Greedy Algorithm

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Summing up over $k = n, \ldots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

Studying Greedy Algorithm

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Summing up over $k = n, \ldots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

Studying Greedy Algorithm

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Summing up over $k = n, \ldots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots =$$

Studying Greedy Algorithm

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

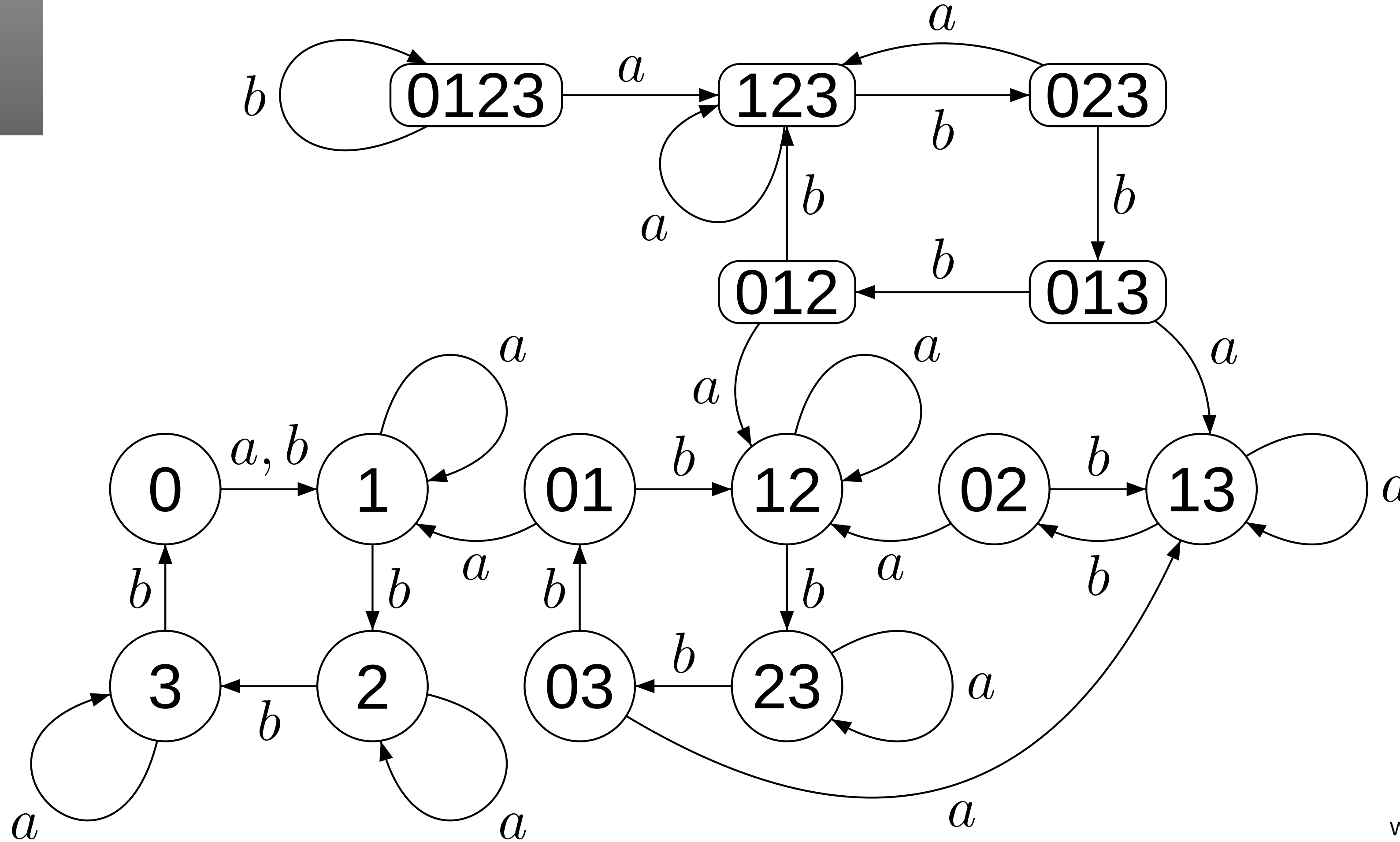
$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots = \binom{n+1}{3} = \frac{n^3 - n}{6}$$

Studying Greedy Algorithm

We have already seen that the greedy algorithm fails to find a reset word of minimum length.

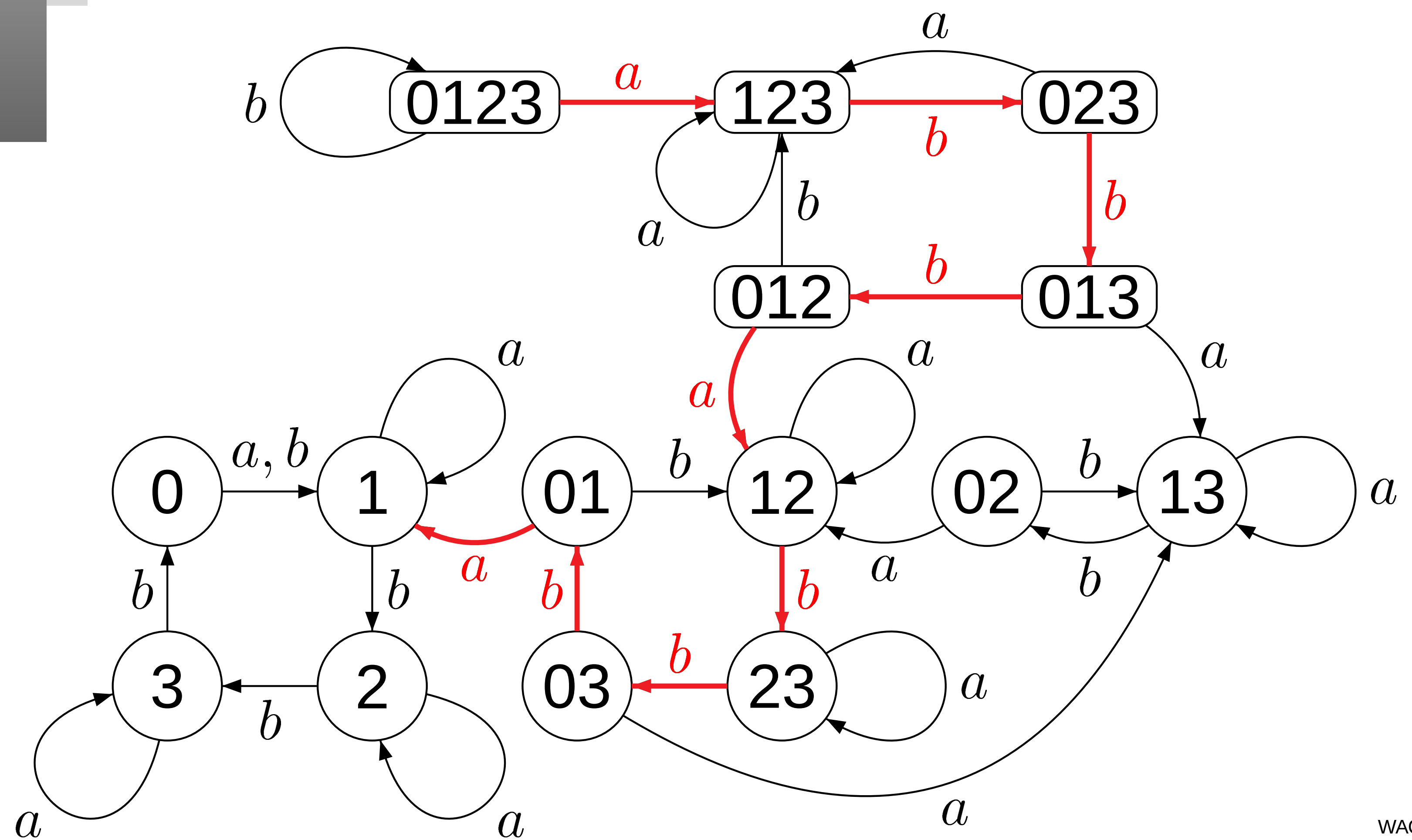
Studying Greedy Algorithm

We have already seen that the greedy algorithm fails to find a reset word of minimum length.



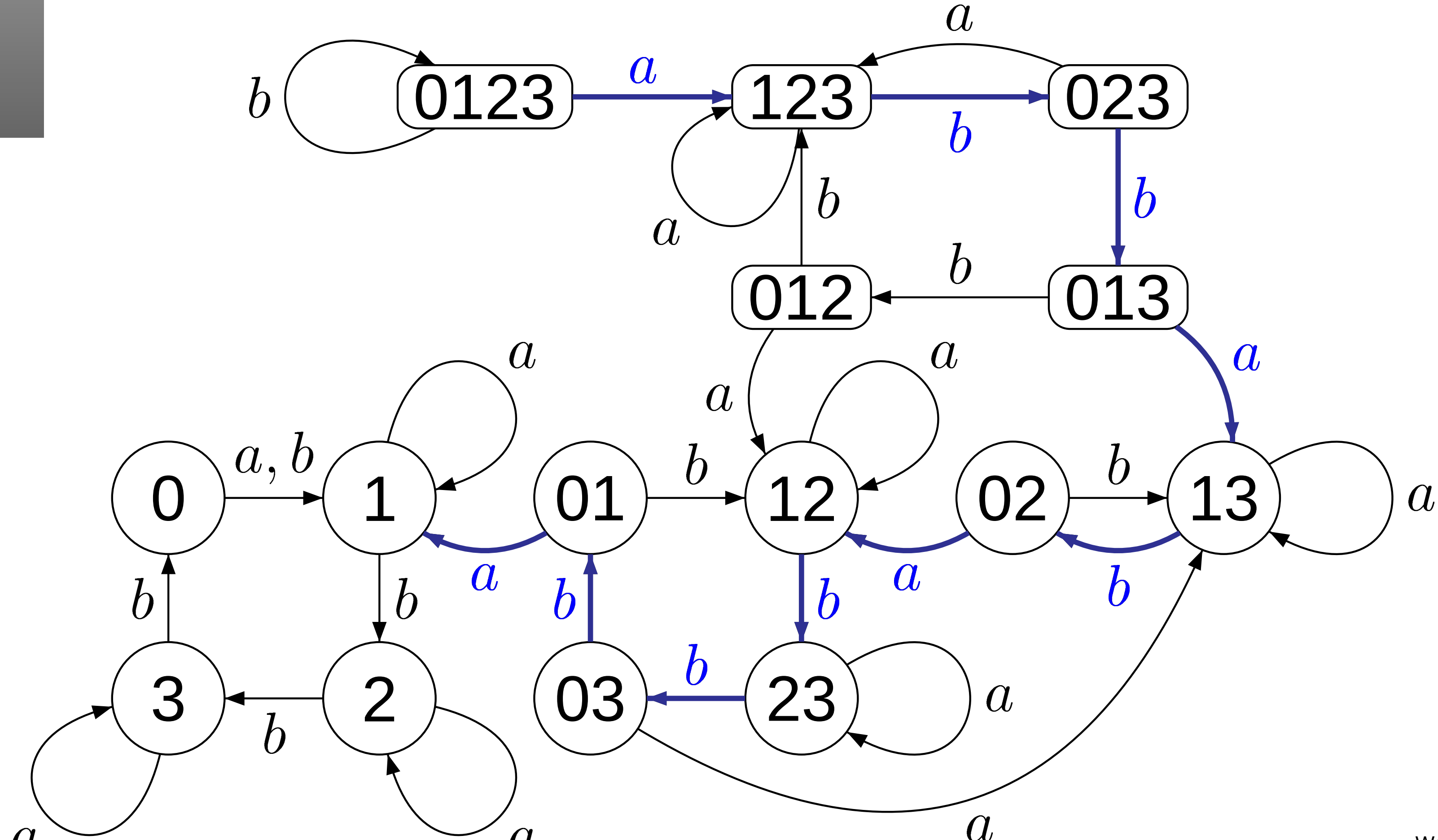
Studying Greedy Algorithm

We have already seen that the greedy algorithm fails to find a reset word of minimum length.



Studying Greedy Algorithm

We have already seen that the greedy algorithm fails to find a reset word of minimum length.



Studying Greedy Algorithm

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large:

Studying Greedy Algorithm

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

Studying Greedy Algorithm

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood;

Studying Greedy Algorithm

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Studying Greedy Algorithm

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Under standard assumptions (like NP $\neq$ coNP) no polynomial algorithm, even non-deterministic, can find the minimum length of reset words for synchronizing automata, see the proceedings.

Studying Greedy Algorithm

However, all known results were consistent with the existence of very good polynomial approximation algorithms for the problem!

Studying Greedy Algorithm

However, all known results were consistent with the existence of very good polynomial approximation algorithms for the problem!

Very recently, Mikhail Berlinkov. a PhD student of mine, has shown that under NP $\neq$ P, for no k, there may exists a polynomial algorithm that, given a synchronizing automaton, produces a reset word whose length is less than $k\times$minimum possible length of a reset word.

The Černý automata

Suppose a synchronizing automaton has n states.
What is the length of its shortest reset word?

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word? We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word? We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

In his 1964 paper Jan Černý constructed a series $\mathscr{C}_n$, $n = 2, 3, \ldots$, of synchronizing automata over 2 letters.

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word?
We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?
In his 1964 paper Jan Černý constructed a series $\mathscr{C}_n$, $n = 2, 3, \dots$, of synchronizing automata over 2 letters. The states of $\mathscr{C}_n$ are the residues modulo n, and the input letters a and b act as follows:

$\delta(0, a) = 1,\ \delta(m, a) = m$ for $0 < m < n$, $\delta(m, b) = m+1 \pmod{n}$

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word?

We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

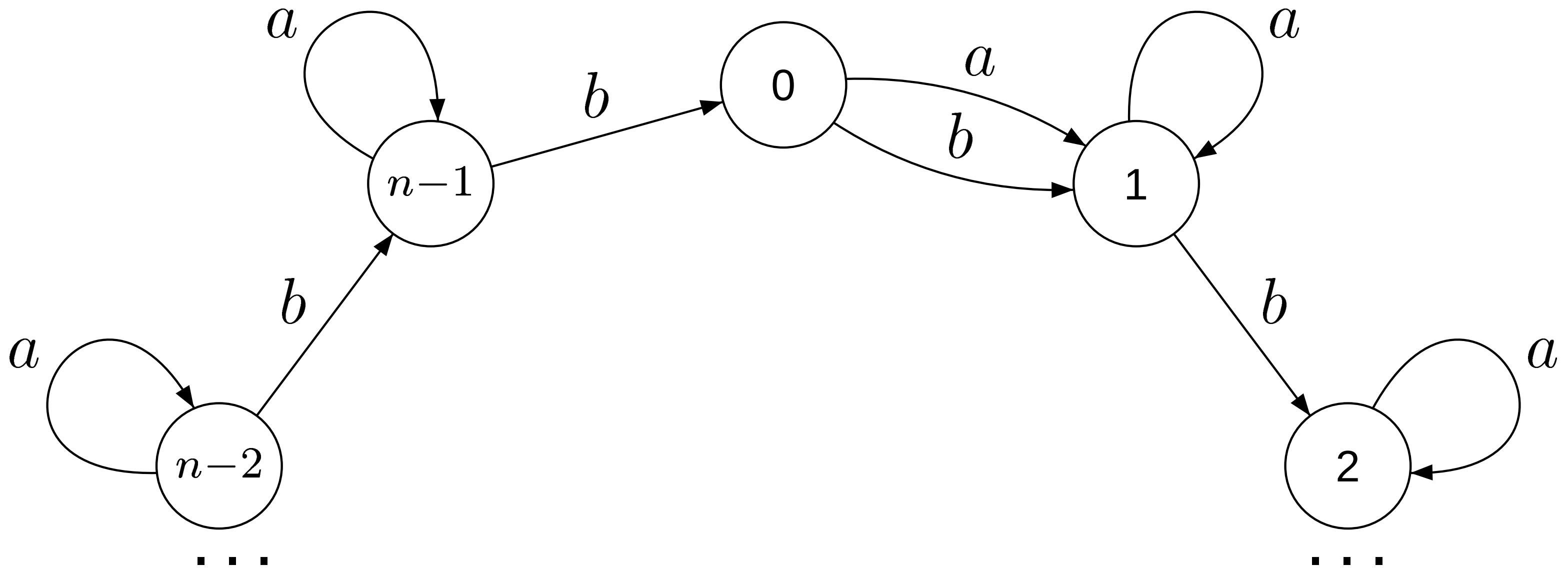
In his 1964 paper Jan Černý constructed a series $\mathscr{C}_n$, $n = 2, 3, \ldots$, of synchronizing automata over 2 letters. The states of $\mathscr{C}_n$ are the residues modulo n, and the input letters a and b act as follows:

$\delta(0, a) = 1,\ \delta(m, a) = m$ for $0 < m < n$, $\delta(m, b) = m+1 \pmod n$

The automaton in the previous slide is $\mathscr{C}_4$.

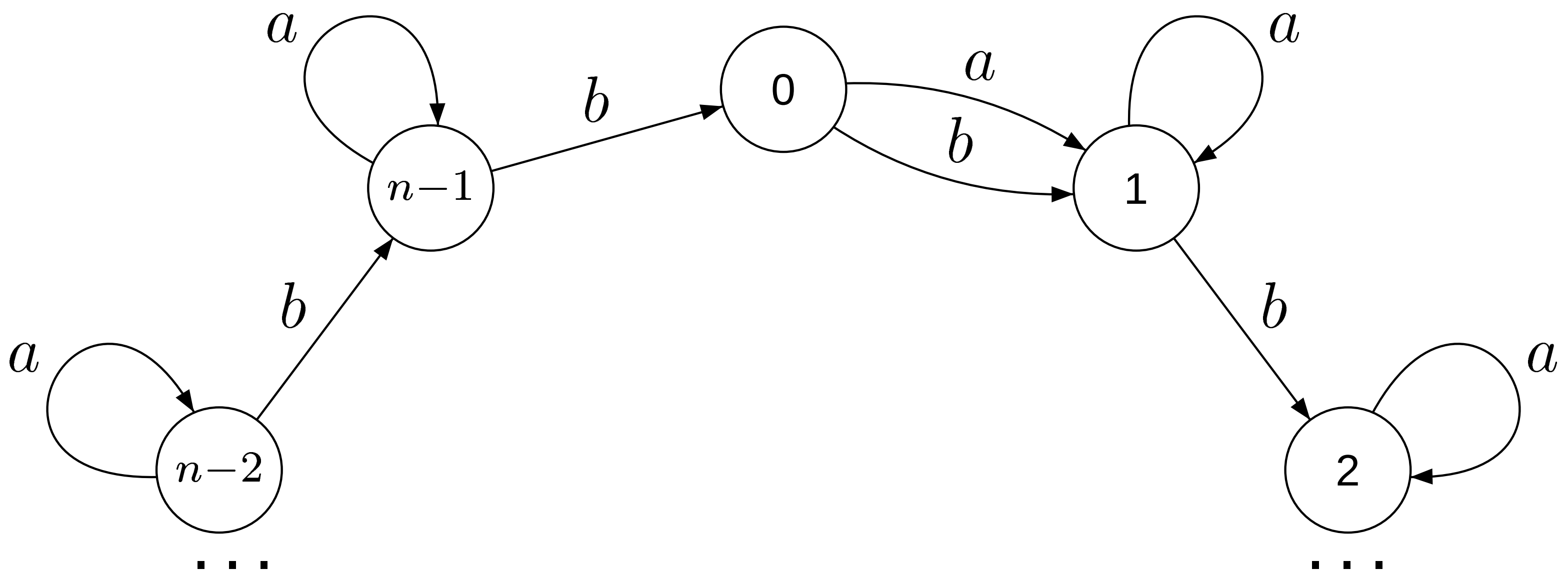
The Černý automata

Here is a generic automaton from the Černý series:



The Černý automata

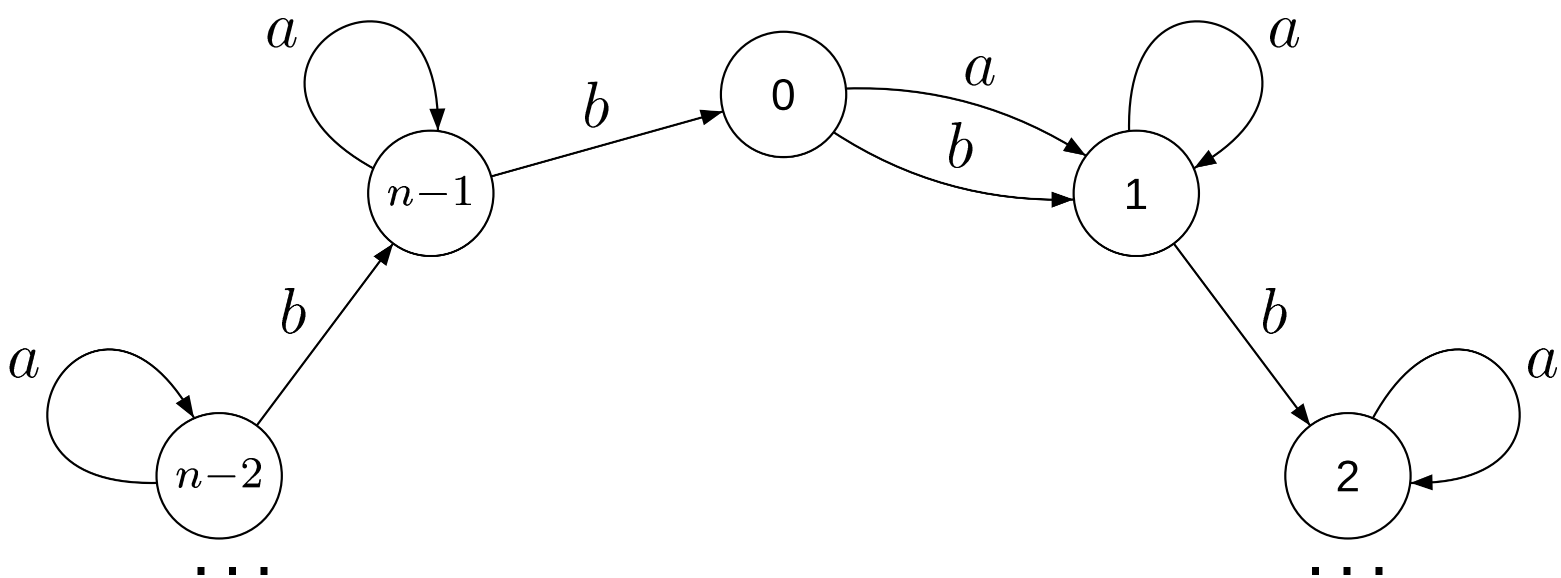
Here is a generic automaton from the Černý series:



Černý has proved that the shortest reset word for $\mathscr{C}_n$ is $(ab^{n-1})^{n-2}a$ of length $(n-1)^2$.

The Černý automata

Here is a generic automaton from the Černý series:



Černý has proved that the shortest reset word for $\mathscr{C}_n$ is $(ab^{n-1})^{n-2}a$ of length $(n-1)^2$. As other results from Černý's paper of 1964, this nice series of automata has been rediscovered many times, see references in the pre-proceedings.

The Černý automata

We present a proof of this result using a solitaire-like game:

We present a proof of this result using a solitaire-like game:

- The digraph of $\mathscr{C}_n$ — the game-board.

The Černý automata

We present a proof of this result using a solitaire-like game:

- The digraph of $\mathscr{C}_n$ — the game-board.
- The initial position — each state holds a coin, all coins are pairwise distinct.

The Černý automata

We present a proof of this result using a solitaire-like game:

- The digraph of $\mathscr{C}_n$ — the game-board.
- The initial position — each state holds a coin, all coins are pairwise distinct.
- Each letter $c \in \{a, b\}$ defines a move — coins slide along the arrows labelled c and, whenever two coins meet at the state 1, the coin arriving from 0 is removed.

The Černý automata

We present a proof of this result using a solitaire-like game:

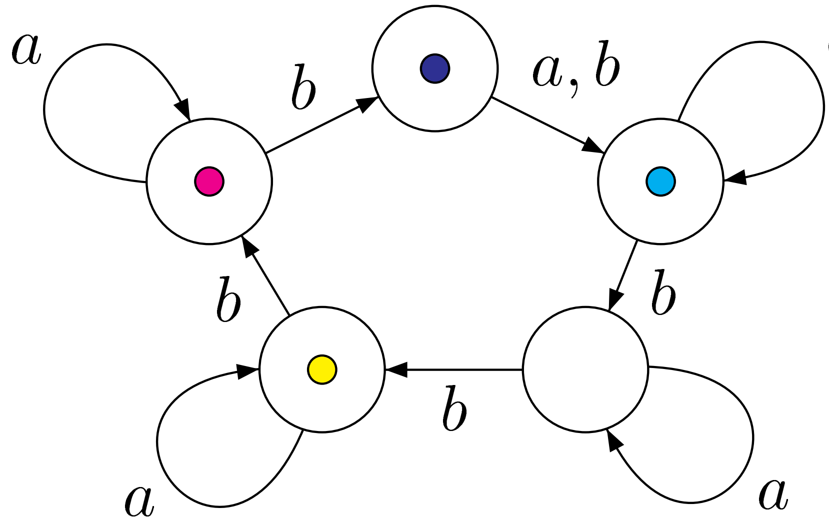
- The digraph of $\mathscr{C}_n$ — the game-board.
- The initial position — each state holds a coin, all coins are pairwise distinct.
- Each letter $c \in \{a, b\}$ defines a move — coins slide along the arrows labelled c and, whenever two coins meet at the state 1, the coin arriving from 0 is removed.
- The goal — to free all but one states.

The Černý automata

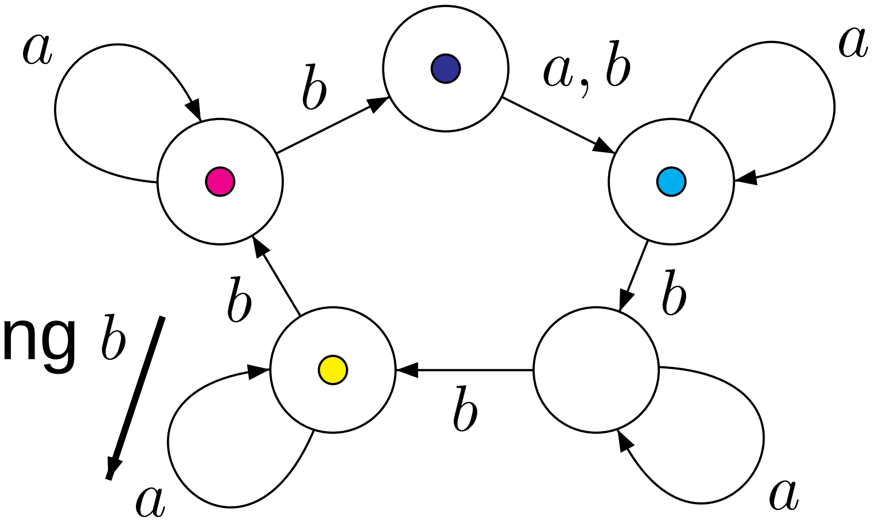
We present a proof of this result using a solitaire-like game:

- The digraph of $\mathscr{C}_n$ — the game-board.
- The initial position — each state holds a coin, all coins are pairwise distinct.
- Each letter $c \in \{a, b\}$ defines a move — coins slide along the arrows labelled c and, whenever two coins meet at the state 1, the coin arriving from 0 is removed.
- The goal — to free all but one states.
- The only coin that remains at the end of the game is the golden coin G.

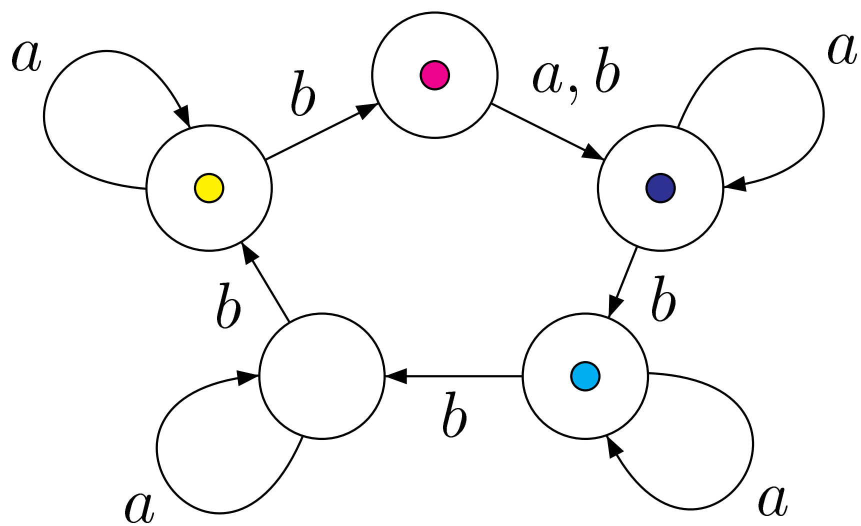
The Černý automata



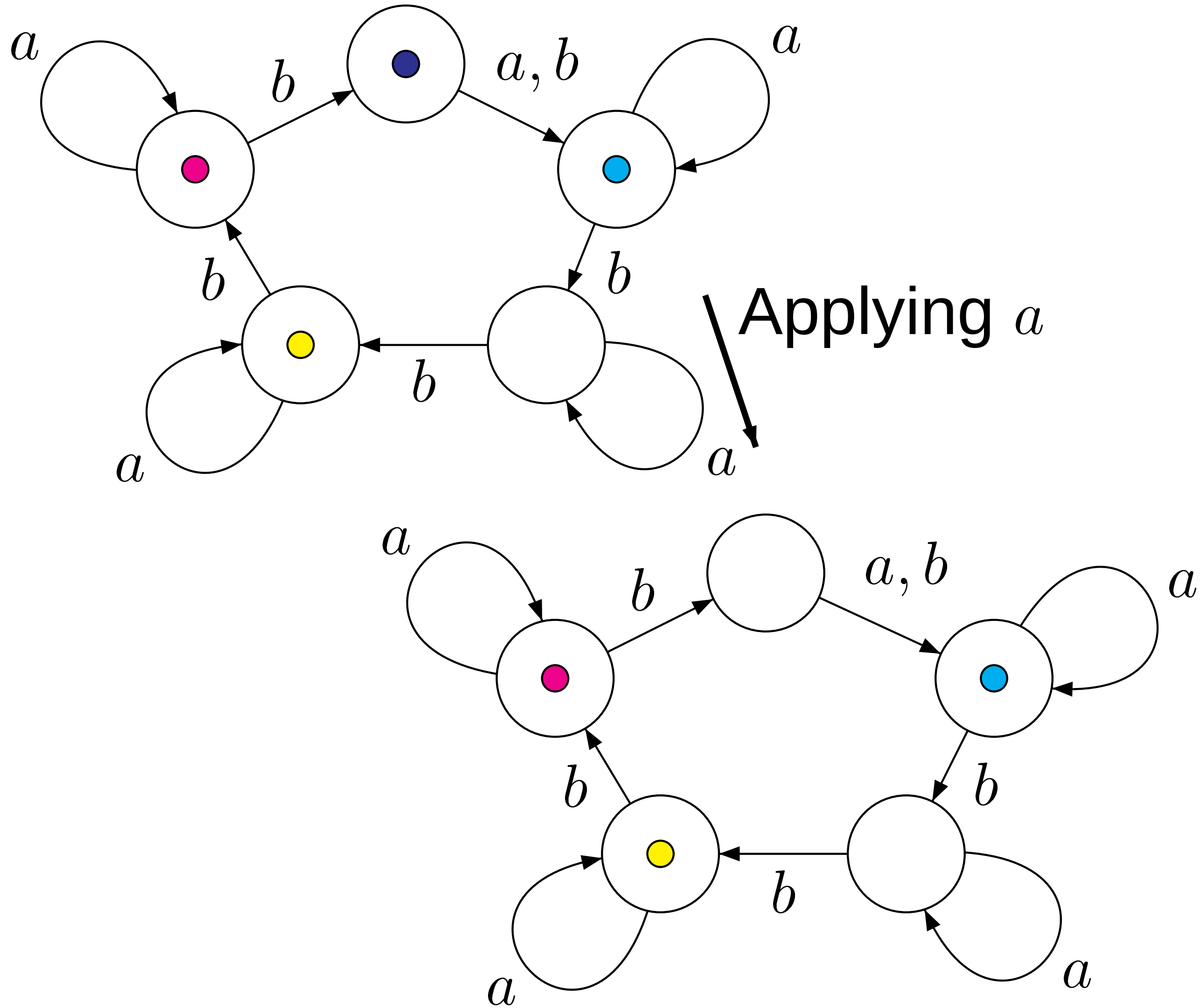
The Černý automata



Applying b



The Černý automata



The Černý automata

Let P_0 be an initial distribution of coins, w a reset word.

The Černý automata

Let P_0 be an initial distribution of coins, w a reset word. Denote by P_i the position that arises when we apply the prefix of w of length i to the position P_0.

The Černý automata

Let P_0 be an initial distribution of coins, w a reset word. Denote by P_i the position that arises when we apply the prefix of w of length i to the position P_0. We want to define the weight $\mathrm{wg}(P_i)$ of the position such that

The Černý automata

Let P_0 be an initial distribution of coins, w a reset word. Denote by P_i the position that arises when we apply the prefix of w of length i to the position P_0. We want to define the weight $\mathrm{wg}(P_i)$ of the position such that

(i) $\mathrm{wg}(P_0) \geq n(n-1)$ and $\mathrm{wg}(P_{|w|}) \leq n-1$;

(ii) for each $i = 1, \ldots, |w|$, the action of the i^{th} letter of w decreases the weight by 1 at most, that is, $1 \geq \mathrm{wg}(P_{i-1}) - \mathrm{wg}(P_i)$.

Let P_0 be an initial distribution of coins, w a reset word. Denote by P_i the position that arises when we apply the prefix of w of length i to the position P_0. We want to define the weight $\mathrm{wg}(P_i)$ of the position such that

(i) $\mathrm{wg}(P_0) \geq n(n-1)$ and $\mathrm{wg}(P_{|w|}) \leq n-1$;

(ii) for each $i = 1, \ldots, |w|$, the action of the i^{th} letter of w decreases the weight by 1 at most, that is, $1 \geq \mathrm{wg}(P_{i-1}) - \mathrm{wg}(P_i)$.

Then $|w| = \sum_{i=1}^{|w|} 1 \geq \sum_{i=1}^{|w|} \big(\mathrm{wg}(P_{i-1}) - \mathrm{wg}(P_i)\big) =$

$$\mathrm{wg}(P_0) - \mathrm{wg}(P_{|w|}) \geq n(n-1) - (n-1) = (n-1)^2.$$

The Černý automata

The trick consists in letting the weight of each coin depend on its relative location w.r.t. the golden coin.

The Černý automata

The trick consists in letting the weight of each coin depend on its relative location w.r.t. the golden coin. If a coin C is present in a position P_i, let $s_i(C)$ be the state covered with C in this position. We define the *weight of C in the position P_i* as

$$\mathrm{wg}(C, P_i) = n \cdot d_i(C) + m_i(C)$$

where $m_i(C)$ is the residue of $n - s_i(C)$ modulo n and $d_i(C)$ is the number of steps from $s_i(C)$ to $s_i(G)$ in the 'main circle' of our automaton. (Recall that G stands for the golden coin G which is present in all positions.)

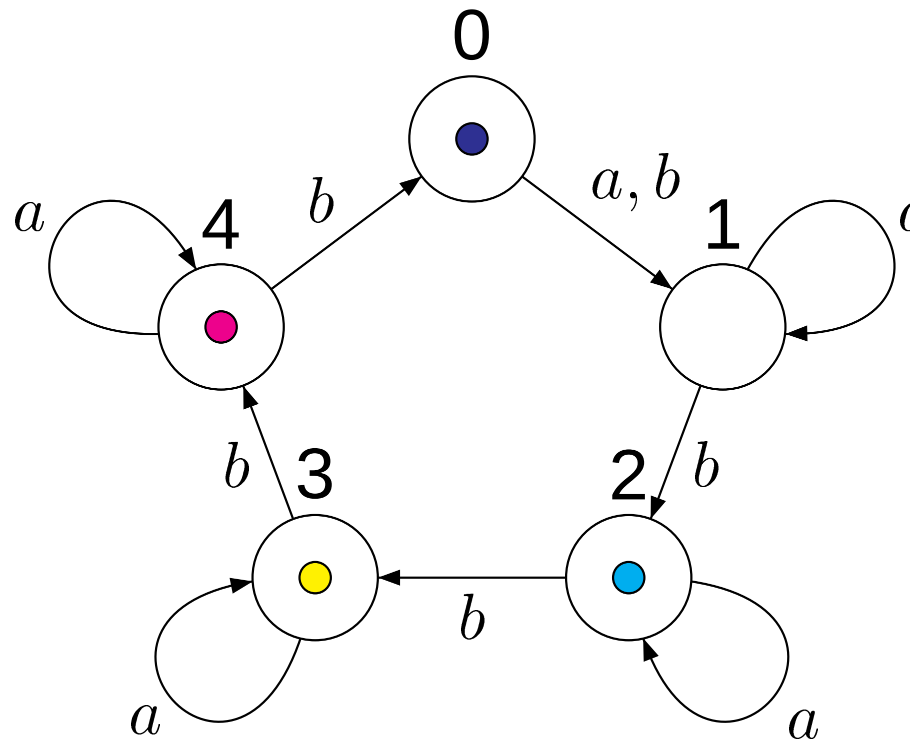
The Černý automata

The trick consists in letting the weight of each coin depend on its relative location w.r.t. the golden coin. If a coin C is present in a position P_i, let $s_i(C)$ be the state covered with C in this position. We define the *weight of C in the position P_i* as

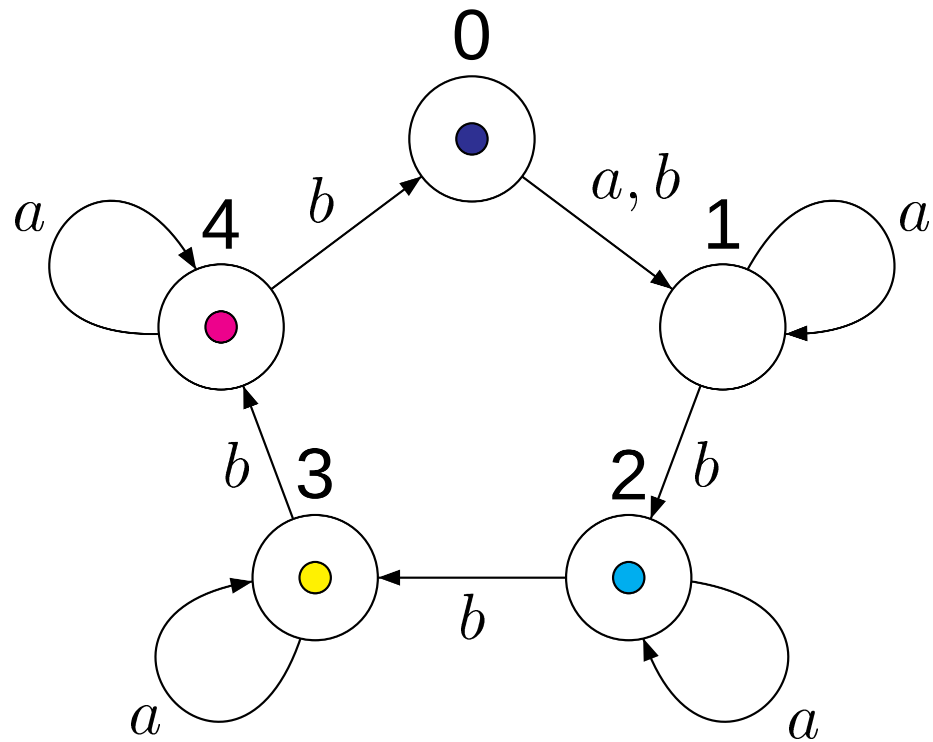
$$\mathrm{wg}(C, P_i) = n \cdot d_i(C) + m_i(C)$$

where $m_i(C)$ is the residue of $n - s_i(C)$ modulo n and $d_i(C)$ is the number of steps from $s_i(C)$ to $s_i(G)$ in the 'main circle' of our automaton. (Recall that G stands for the golden coin G which is present in all positions.) The weight of P_i is the maximum weight of the coins present in this position.

The Černý automata

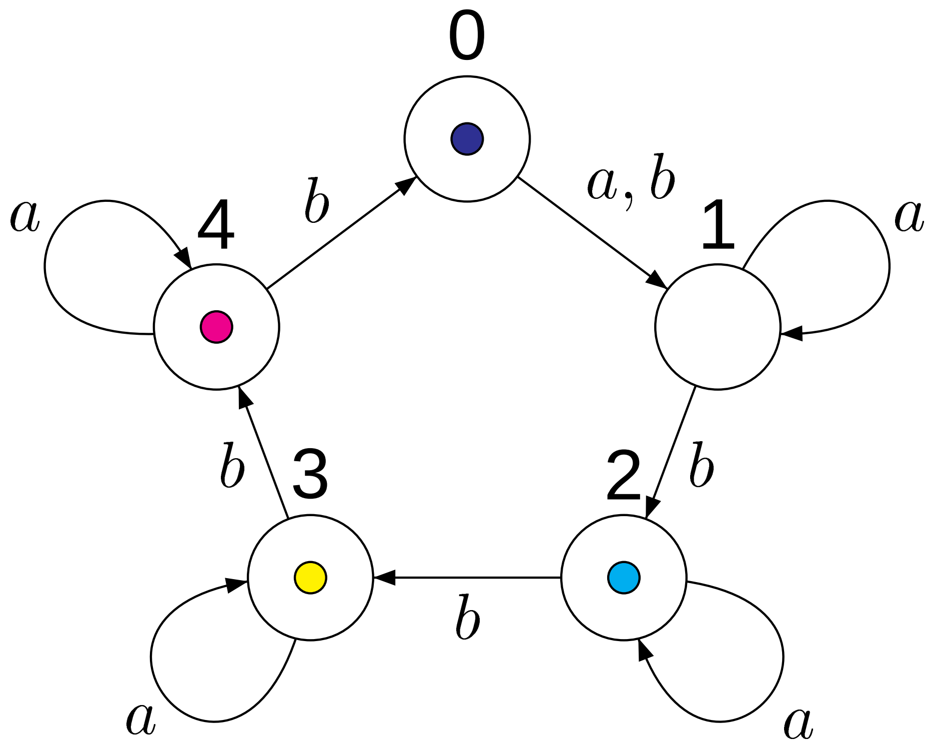


The Černý automata



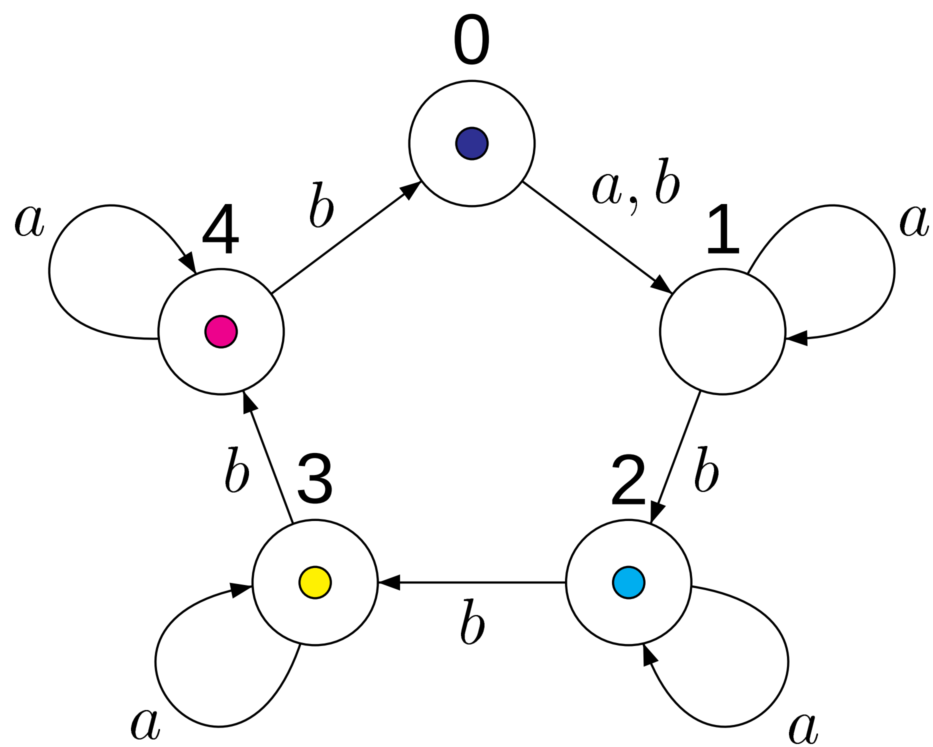
Assume that the yellow coin is the golden one.

The Černý automata



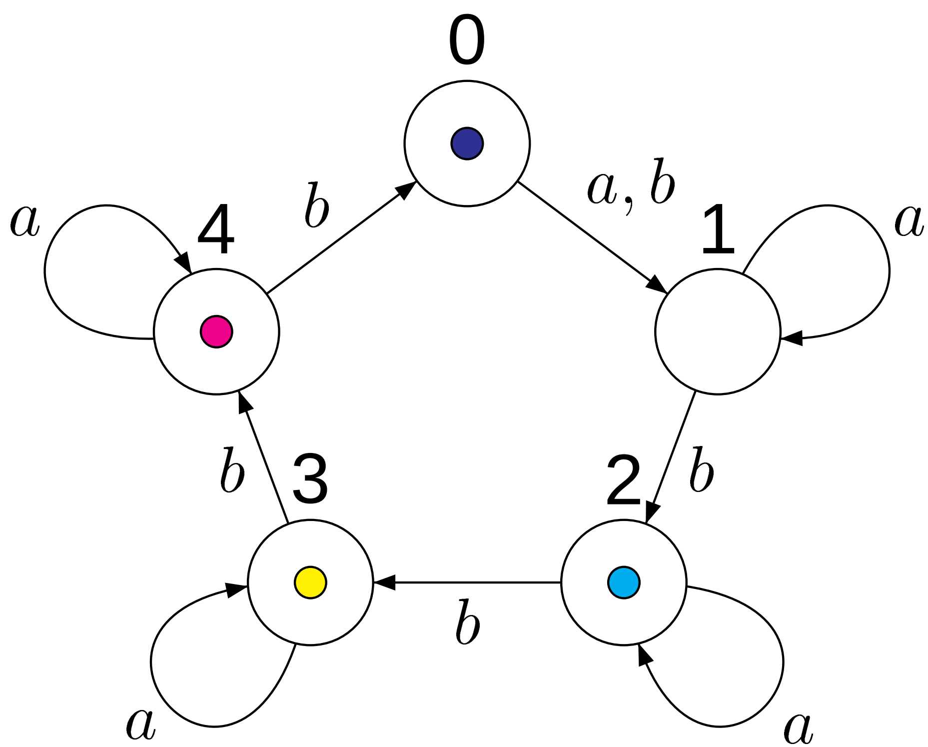
Assume that the yellow coin is the golden one. Then its weight is $5 - 3 = 2$.

The Černý automata



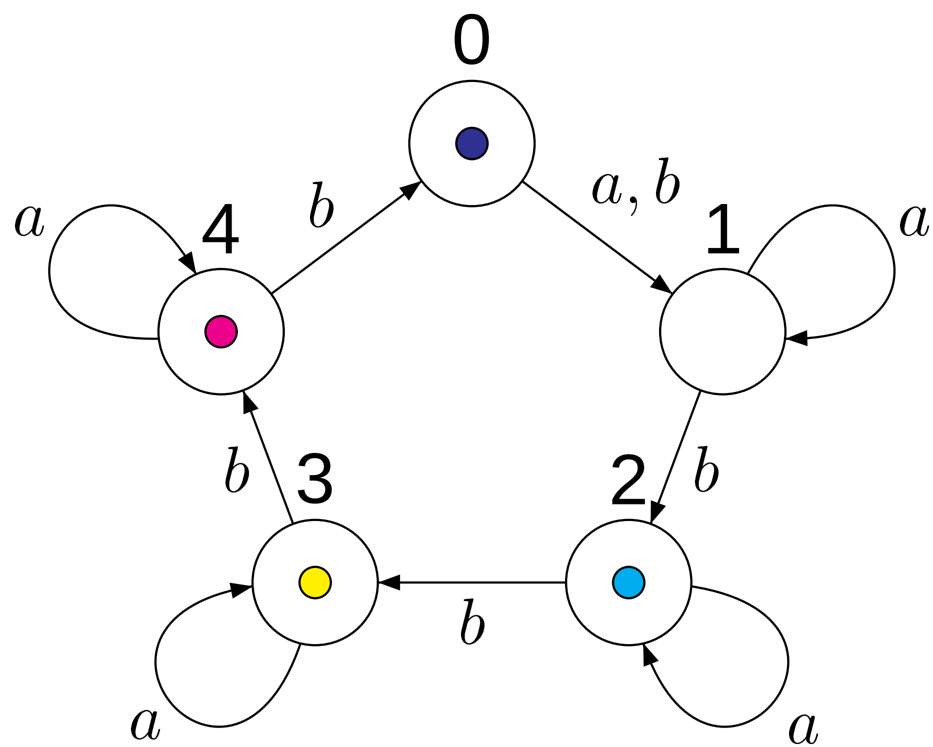
Assume that the yellow coin is the golden one. Then its weight is $5 - 3 = 2$. The weight of the cyan coin is $5 \cdot 1 + (5 - 2) = 8$.

The Černý automata



Assume that the yellow coin is the golden one. Then its weight is $5 - 3 = 2$. The weight of the cyan coin is $5 \cdot 1 + (5 - 2) = 8$. The weight of the blue coin is $5 \cdot 3 + 0 = 15$.

The Černý automata



Assume that the yellow coin is the golden one. Then its weight is $5 - 3 = 2$. The weight of the cyan coin is $5 \cdot 1 + (5 - 2) = 8$. The weight of the blue coin is $5 \cdot 3 + 0 = 15$. The weight of the magenta coin is $5 \cdot 4 + (5 - 4) = 21$, and this is the weight of the position.

The Černý automata

We have to check that our weight function satisfies the conditions

(i) $\mathrm{wg}(P_0) \geq n(n-1)$ and $\mathrm{wg}(P_{|w|}) \leq n-1$;

(ii) $1 \geq \mathrm{wg}(P_{i-1}) - \mathrm{wg}(P_i)$ for each $i = 1, \ldots, |w|$.

The Černý automata

We have to check that our weight function satisfies the conditions

(i) $\mathrm{wg}(P_0) \geq n(n-1)$ and $\mathrm{wg}(P_{|w|}) \leq n-1$;

(ii) $1 \geq \mathrm{wg}(P_{i-1}) - \mathrm{wg}(P_i)$ for each $i = 1, \ldots, |w|$.

In the initial position all states are covered with coins. Consider the coin C that covers the state $s_0(G) + 1 \pmod{n}$, that is the state in one step clockwise after the state covered with the golden coin. Then $d_0(C) = n - 1$ whence $\mathrm{wg}(C, P_0) = n \cdot (n-1) + m_0(C) \geq n(n-1)$. Since the weight of a position is not less that the weight of any coin in this position, we have $\mathrm{wg}(P_0) \geq n(n-1)$.

The Černý automata

In the final position only the golden coin G remains, whence the weight of $P_{|w|}$ is the weight of G. Clearly, $\mathrm{wg}(G, P_i) = m_i(G) \leq n - 1$ for any position P_i.

The Černý automata

In the final position only the golden coin G remains, whence the weight of $P_{|w|}$ is the weight of G. Clearly, $\mathrm{wg}(G, P_i) = m_i(G) \leq n - 1$ for any position P_i.

Let C be a coin of maximum weight in P_{i-1}. If the transition from P_{i-1} to P_i is caused by b, then $d_i(C) = d_{i-1}(C)$ (because the relative location of the coins does not change) and $m_i(C) = m_{i-1}(C) - 1$ if $m_{i-1}(C) > 0$, otherwise $m_i(C) = n - 1$.

The Černý automata

In the final position only the golden coin G remains, whence the weight of $P_{|w|}$ is the weight of G. Clearly, $\mathrm{wg}(G, P_i) = m_i(G) \leq n - 1$ for any position P_i.

Let C be a coin of maximum weight in P_{i-1}. If the transition from P_{i-1} to P_i is caused by b, then $d_i(C) = d_{i-1}(C)$ (because the relative location of the coins does not change) and $m_i(C) = m_{i-1}(C) - 1$ if $m_{i-1}(C) > 0$, otherwise $m_i(C) = n - 1$. We see that

$$\mathrm{wg}(P_i) \geq \mathrm{wg}(C, P_i) = n \cdot d_i(C) + m_i(C) \geq$$

$$n \cdot d_{i-1}(C) + m_{i-1}(C) - 1 = \mathrm{wg}(C, P_{i-1}) - 1 = \mathrm{wg}(P_{i-1}) - 1.$$

The Černý automata

Suppose the transition from P_{i-1} to P_i is caused by a. If $s_{i-1}(C) \neq 0$, then $m_i(C) = m_{i-1}(C)$ and $d_i(C) = d_{i-1}(C)$ if $s_{i-1}(G) \neq 0$, otherwise $d_i(C) = d_{i-1}(C) + 1$.

The Černý automata

Suppose the transition from P_{i-1} to P_i is caused by a. If $s_{i-1}(C) \neq 0$, then $m_i(C) = m_{i-1}(C)$ and $d_i(C) = d_{i-1}(C)$ if $s_{i-1}(G) \neq 0$, otherwise $d_i(C) = d_{i-1}(C) + 1$. Thus, the transition from P_{i-1} to P_i cannot decrease the weight.

The Černý automata

Suppose the transition from P_{i-1} to P_i is caused by a. If $s_{i-1}(C) \neq 0$, then $m_i(C) = m_{i-1}(C)$ and $d_i(C) = d_{i-1}(C)$ if $s_{i-1}(G) \neq 0$, otherwise $d_i(C) = d_{i-1}(C) + 1$. Thus, the transition from P_{i-1} to P_i cannot decrease the weight.

Assume that C covers 0 in P_{i-1}. Then in P_i the state 1 holds a coin C' (which may or may not coincide with C). In P_{i-1} the golden coin G does not cover 0 whence it does not move and $d_i(C') = d_{i-1}(C) - 1$.

The Černý automata

Suppose the transition from P_{i-1} to P_i is caused by a. If $s_{i-1}(C) \neq 0$, then $m_i(C) = m_{i-1}(C)$ and $d_i(C) = d_{i-1}(C)$ if $s_{i-1}(G) \neq 0$, otherwise $d_i(C) = d_{i-1}(C) + 1$. Thus, the transition from P_{i-1} to P_i cannot decrease the weight.

Assume that C covers 0 in P_{i-1}. Then in P_i the state 1 holds a coin C' (which may or may not coincide with C). In P_{i-1} the golden coin G does not cover 0 whence it does not move and $d_i(C') = d_{i-1}(C) - 1$. Therefore

$$\mathrm{wg}(P_i) \geq \mathrm{wg}(C', P_i) = n \cdot d_i(C') + n - 1 = n \cdot (d_{i-1}(C) - 1) + n - 1$$
$$= n \cdot d_{i-1}(C) - 1 = \mathrm{wg}(C, P_{i-1}) - 1 = \mathrm{wg}(P_{i-1}) - 1.$$

The Černý conjecture

Define the *Černý function* $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, $n = 2, 3, \ldots$, yields the inequality $C(n) \geq (n-1)^2$.

The Černý conjecture

Define the *Černý function* $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, $n = 2, 3, \dots$, yields the inequality $C(n) \geq (n-1)^2$.

The Černý conjecture is the claim that in fact the equality $C(n) = (n-1)^2$ holds true.

The Černý conjecture

Define the *Černý function* $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, $n = 2, 3, \dots$, yields the inequality $C(n) \geq (n-1)^2$.

The Černý conjecture is the claim that in fact the equality $C(n) = (n-1)^2$ holds true. This simply looking conjecture is arguably the most longstanding open problem in the combinatorial theory of finite automata, see the pre-proceedings for a discussion of the history of the conjecture.

The Černý conjecture

Define the *Černý function* $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, $n = 2, 3, \ldots$, yields the inequality $C(n) \geq (n-1)^2$.

The Černý conjecture is the claim that in fact the equality $C(n) = (n-1)^2$ holds true. This simply looking conjecture is arguably the most longstanding open problem in the combinatorial theory of finite automata, see the pre-proceedings for a discussion of the history of the conjecture. Everything we know about the conjecture in general can be summarized in one line:

$$(n-1)^2 \leq C(n) \leq \frac{n^3 - n}{6}.$$

The Černý conjecture

Why is the problem so surprisingly difficult?

The Černý conjecture

Why is the problem so surprisingly difficult?

We saw two reasons in Lecture I:

The Černý conjecture

Why is the problem so surprisingly difficult?

We saw two reasons in Lecture I:

• non-locality: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails;

The Černý conjecture

Why is the problem so surprisingly difficult?

We saw two reasons in Lecture I:

- non-locality: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails;
- combinatorics of finite sets is encoded in the problem.

The Černý conjecture

Why is the problem so surprisingly difficult?

We saw two reasons in Lecture I:
- non-locality: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails;
- combinatorics of finite sets is encoded in the problem.

Yet another reason: “slowly” synchronizing automata turn out to be extremely rare.

The Černý conjecture

Why is the problem so surprisingly difficult?

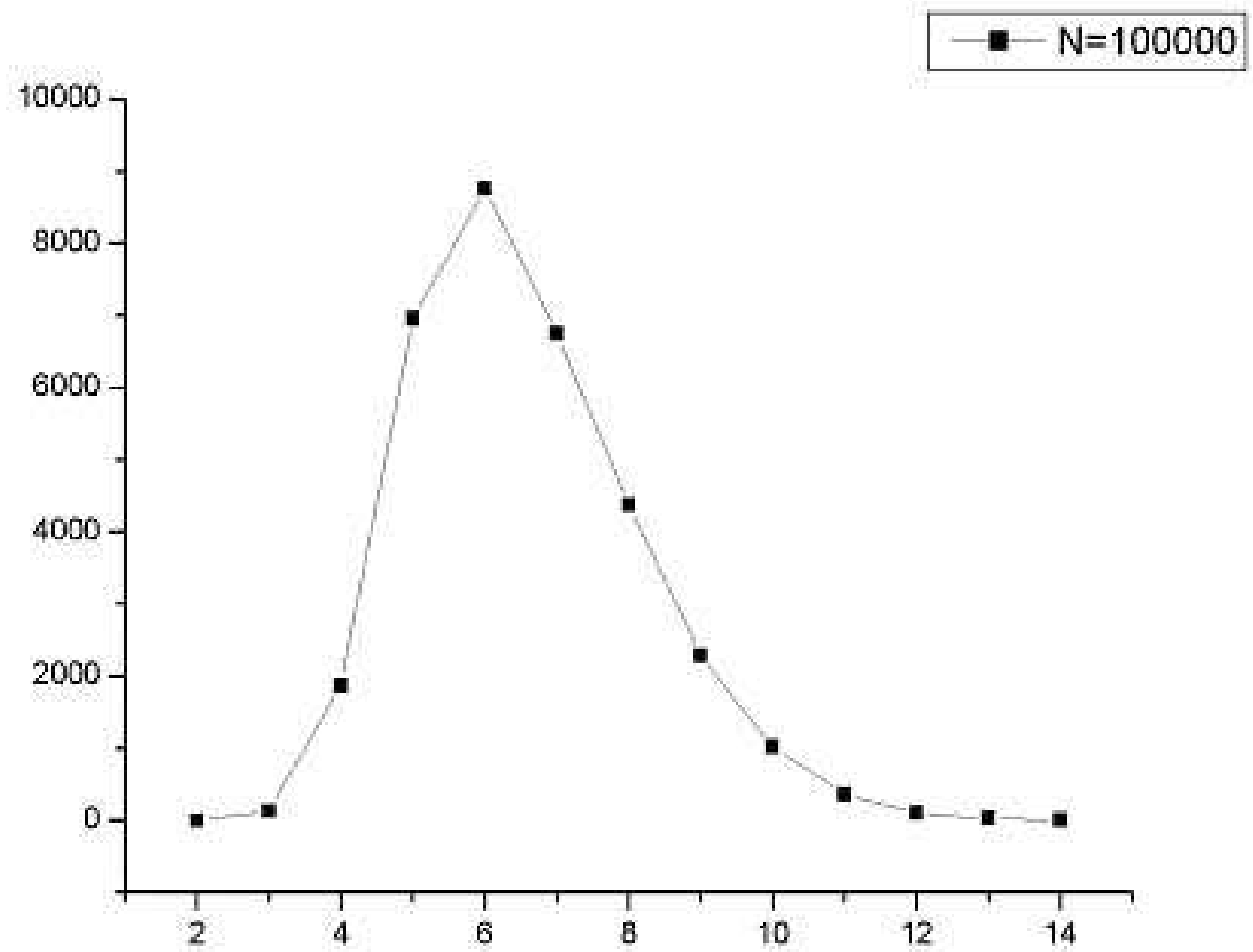
We saw two reasons in Lecture I:

• non-locality: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails;

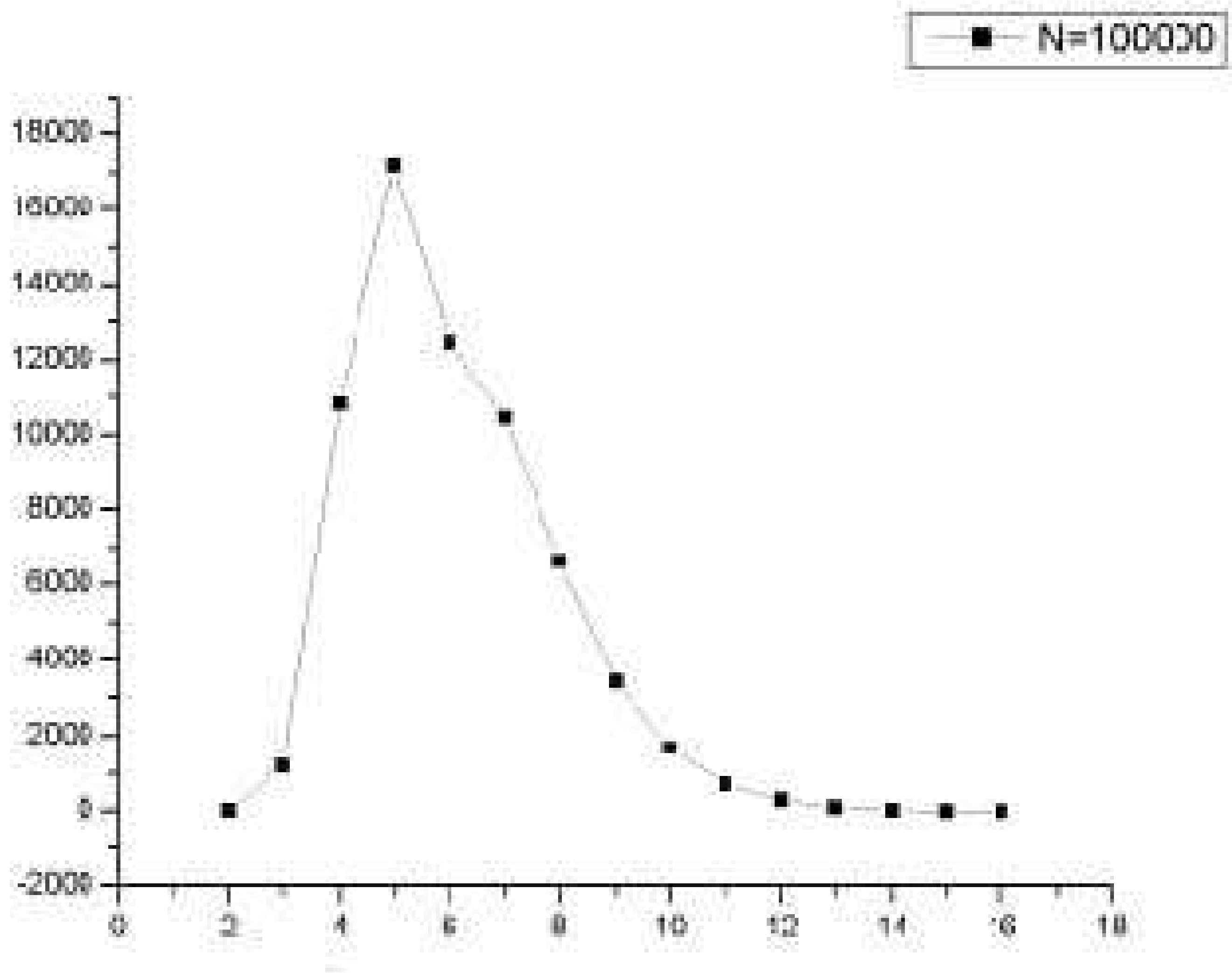
• combinatorics of finite sets is encoded in the problem.

Yet another reason: "slowly" synchronizing automata turn out to be extremely rare. The only known infinite series of n-state synchronizing automata with shortest reset words of length $(n-1)^2$ is the Černý series $\mathscr{C}_n$, $n = 2, 3, \dots$, with a few sporadic examples for $n \leq 6$.

20-State Experiment



30-State Experiment



Random Automata

A (partial) explanation of these experimental observations:

Random Automata

A (partial) explanation of these experimental observations: if Q is an n-set (with n large enough), then, on average, any product of $2n$ randomly chosen transformations of Q is a constant map (Peter Higgins, The range order of a product of i transformations from a finite full transformation semigroup, Semigroup Forum, 37 (1988) 31–36).

Random Automata

A (partial) explanation of these experimental observations: if Q is an n-set (with n large enough), then, on average, any product of $2n$ randomly chosen transformations of Q is a constant map (Peter Higgins, The range order of a product of i transformations from a finite full transformation semigroup, Semigroup Forum, 37 (1988) 31–36). In automata-theoretic terms, this fact means that a randomly chosen DFA with n states and a sufficiently large input alphabet tends to be synchronizing and is reset by any word of length $\leq 2n$.

Random Automata

A (partial) explanation of these experimental observations: if Q is an n-set (with n large enough), then, on average, any product of $2n$ randomly chosen transformations of Q is a constant map (Peter Higgins, The range order of a product of i transformations from a finite full transformation semigroup, Semigroup Forum, 37 (1988) 31–36). In automata-theoretic terms, this fact means that a randomly chosen DFA with n states and a sufficiently large input alphabet tends to be synchronizing and is reset by any word of length $\leq 2n$.

Thus, “slowly” synchronizing automata cannot be discovered via a random sampling.

Sporadic Examples

A synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is *proper* if none of the automata obtained from $\mathscr{A}$ by erasing any letter in Σ are synchronizing.

Sporadic Examples

A synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is *proper* if none of the automata obtained from $\mathscr{A}$ by erasing any letter in Σ are synchronizing. E.g., the Černý automata $\mathscr{C}_n$ with $n > 2$ are proper while $\mathscr{C}_2$ is not.

Sporadic Examples

A synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from $\mathscr{A}$ by erasing any letter in Σ are synchronizing. E.g., the Černý automata $\mathscr{C}_n$ with $n > 2$ are proper while $\mathscr{C}_2$ is not.
A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n-1)^2$.

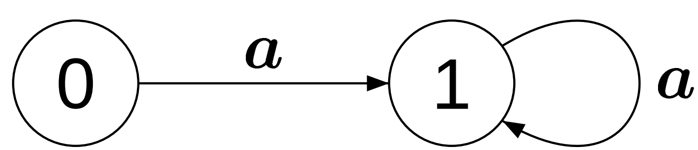
Sporadic Examples

A synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from $\mathscr{A}$ by erasing any letter in Σ are synchronizing. E.g., the Černý automata $\mathscr{C}_n$ with $n > 2$ are proper while $\mathscr{C}_2$ is not.
A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n-1)^2$. We present here all known proper synchronizing automata beyond the Černý series $\mathscr{C}_n$, $n = 3, 4, \ldots$, that reach the Černý bound.

Sporadic Examples

A synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from $\mathscr{A}$ by erasing any letter in Σ are synchronizing. E.g., the Černý automata $\mathscr{C}_n$ with $n > 2$ are proper while $\mathscr{C}_2$ is not. A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n-1)^2$. We present here all known proper synchronizing automata beyond the Černý series $\mathscr{C}_n$, $n = 3, 4, \dots$, that reach the Černý bound.

For the sake of completeness, we start with $n = 2$:

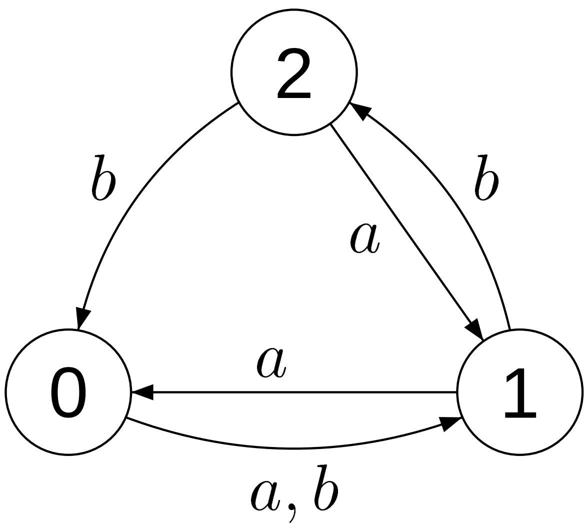


Sporadic Examples

For $n = 3$ we have three sporadic automata:

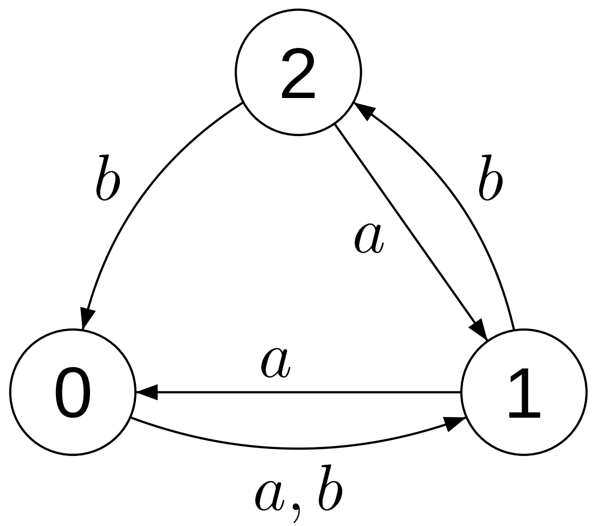
Sporadic Examples

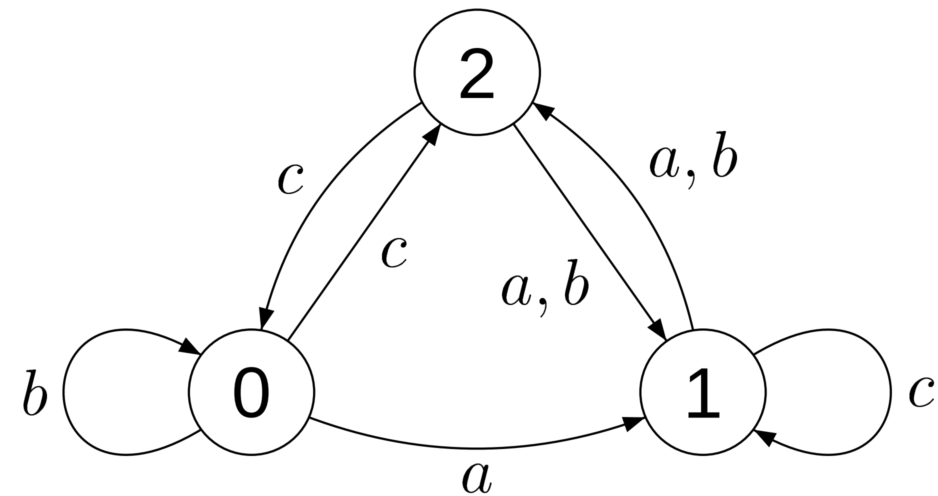
For $n = 3$ we have three sporadic automata:



Sporadic Examples

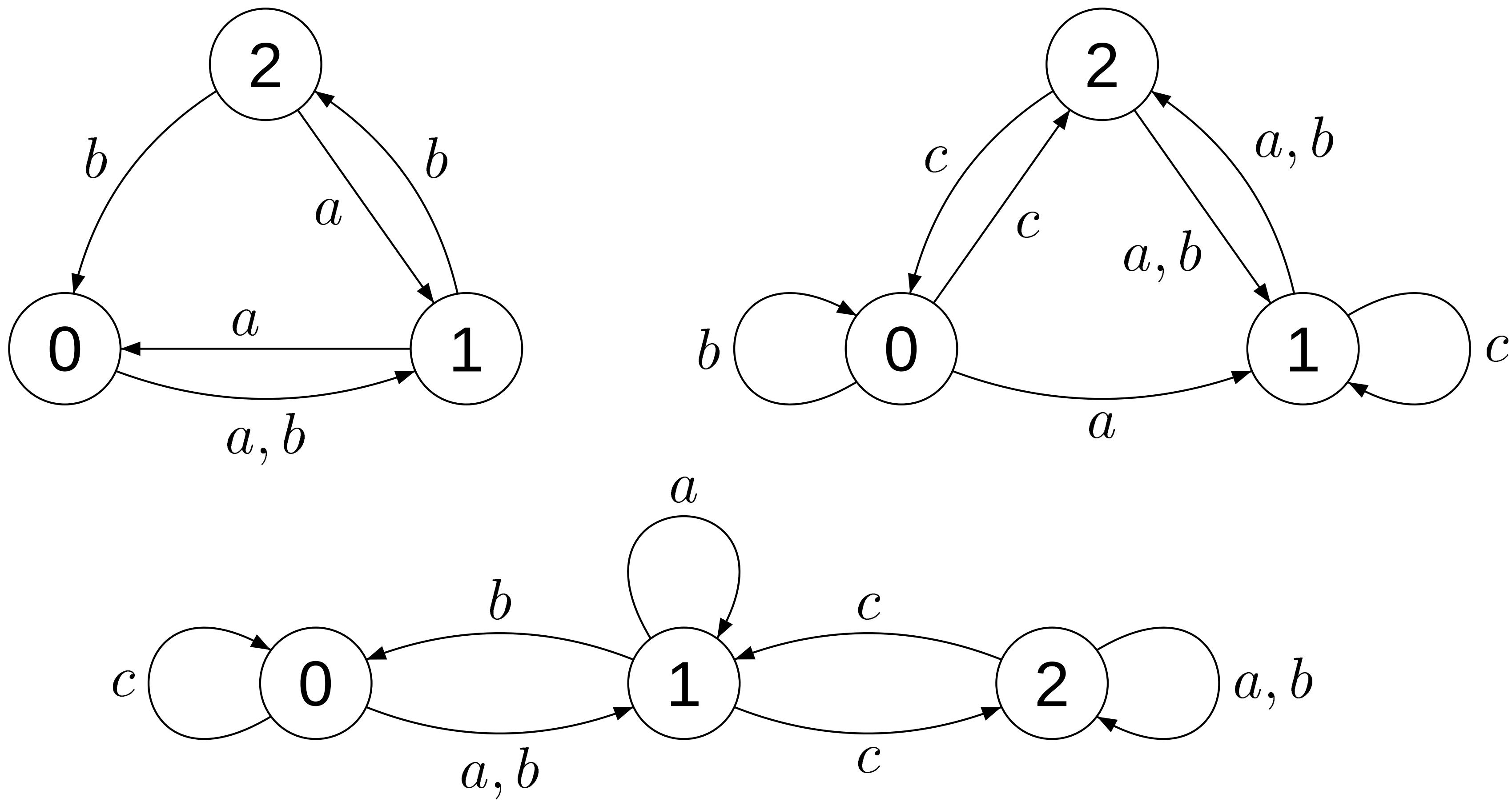
For $n = 3$ we have three sporadic automata:





Sporadic Examples

For $n = 3$ we have three sporadic automata:

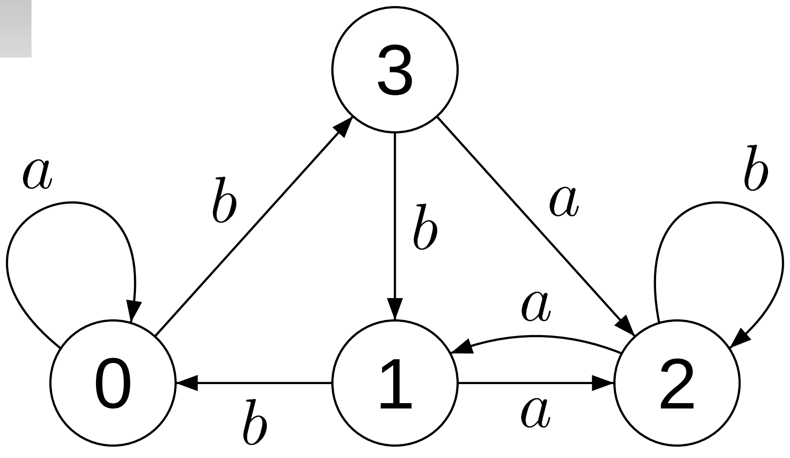


Sporadic Examples

Also for $n = 4$ three sporadic automata are known:

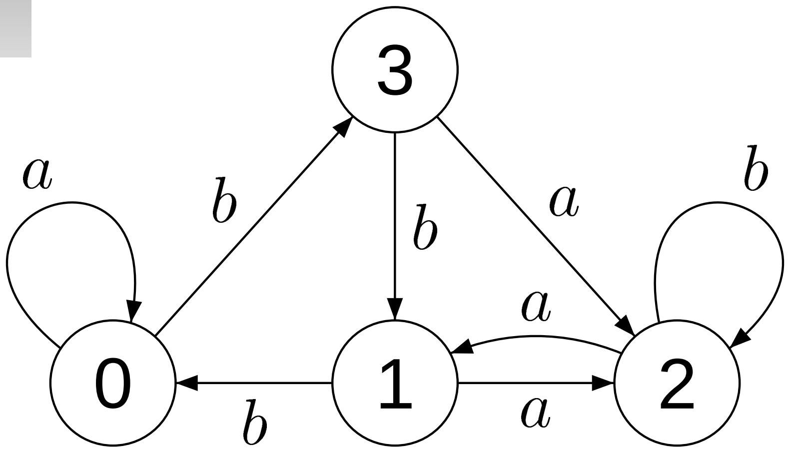
Sporadic Examples

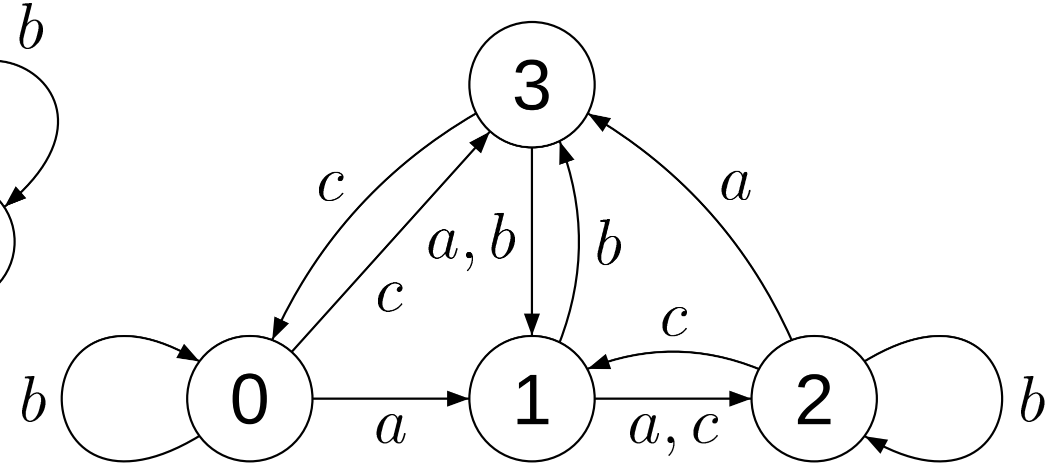
Also for $n = 4$ three sporadic automata are known:



Sporadic Examples

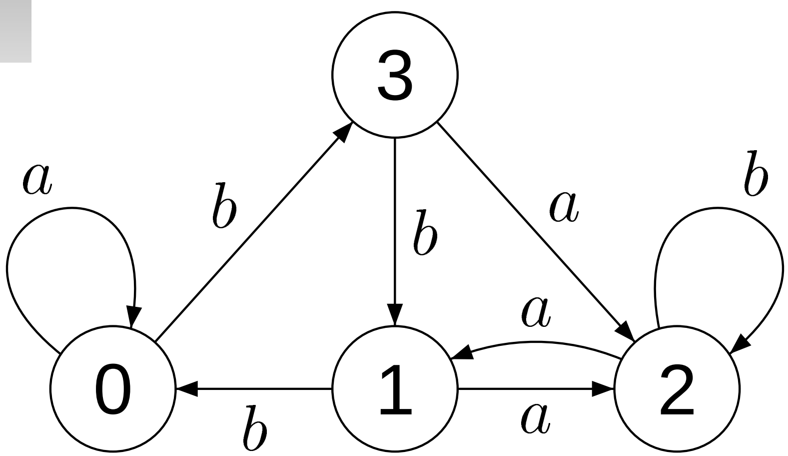
Also for $n = 4$ three sporadic automata are known:

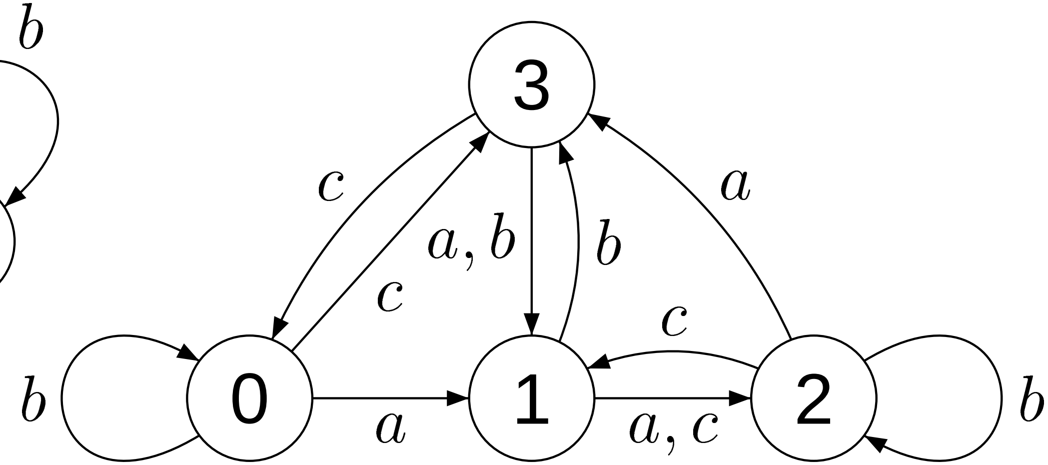


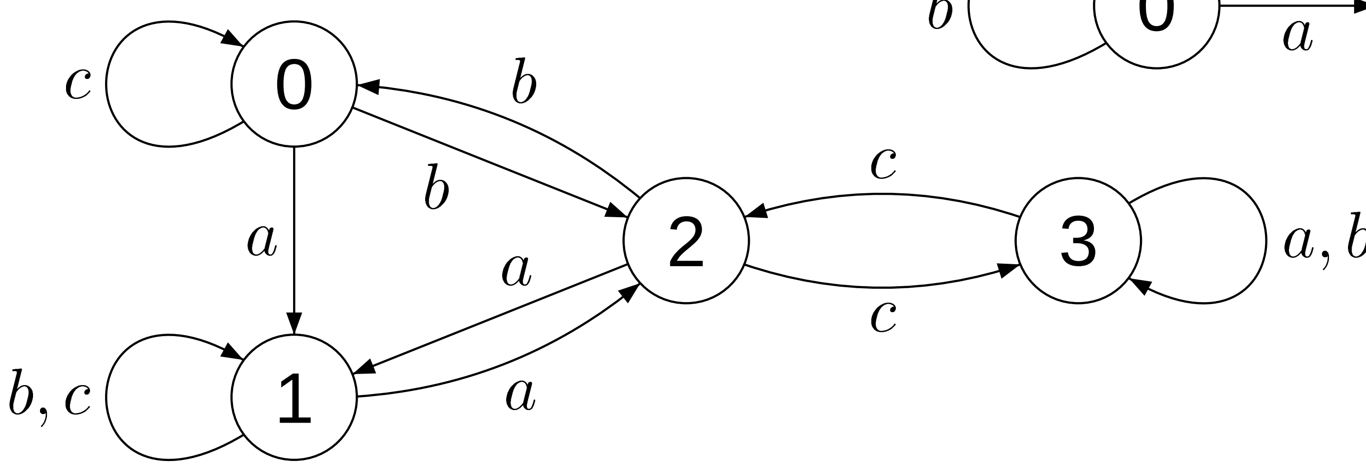


Sporadic Examples

Also for $n = 4$ three sporadic automata are known:





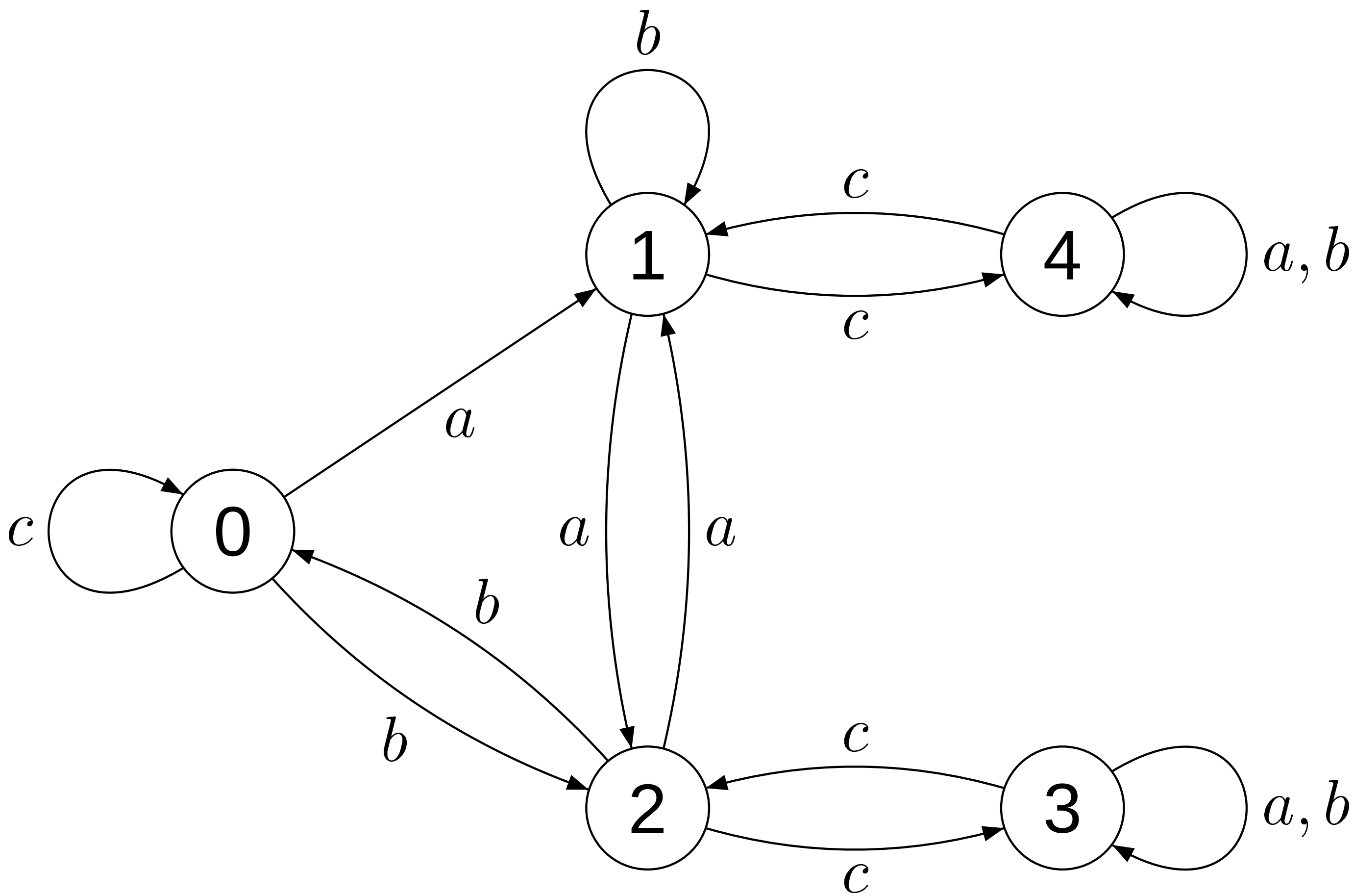


Sporadic Examples

A proper 5-state automaton reaching the Černý bound has been recently discovered by Adam Roman.

Sporadic Examples

A proper 5-state automaton reaching the Černý bound has been recently discovered by Adam Roman.

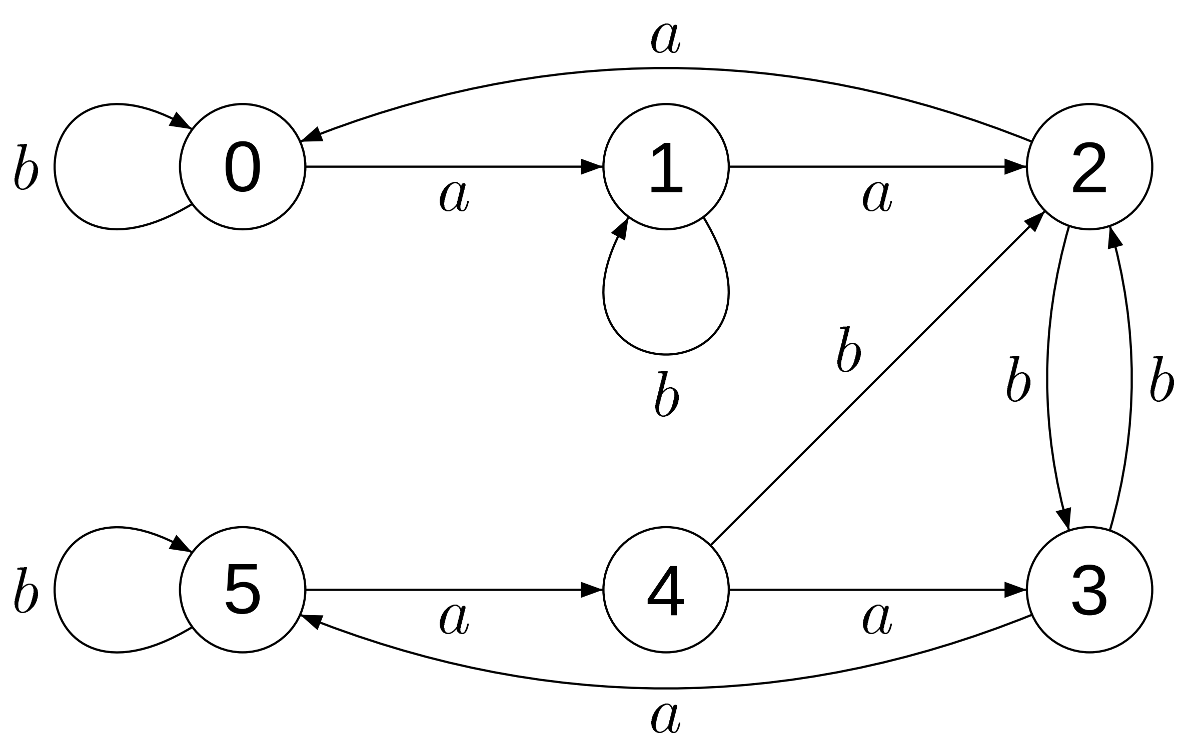


Kari’s Example

The last in our list and the most remarkable example was published in 2001 by Jarkko Kari (A counter example to a conjecture concerning synchronizing words in finite automata, EATCS Bull., 73, 146).

Kari's Example

The last in our list and the most remarkable example was published in 2001 by Jarkko Kari (A counter example to a conjecture concerning synchronizing words in finite automata, EATCS Bull., 73, 146).



Kari's Example

Kari's automaton $\mathscr{K}_{\mathbf{6}}$ has refuted several conjectures.

Kari's Example

Kari's automaton $\mathscr{K}_6$ has refuted several conjectures.

The most well known of them was suggested by Jean-Eric Pin in 1978. Pin conjectured that if a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with n states admits a word $w \in \Sigma^*$ such that $|Q \,.\, w| = k$, $1 \leq k \leq n$, then $\mathscr{A}$ possesses a word of length at most $(n-k)^2$ with the same property.

Kari's Example

Kari's automaton $\mathscr{K}_6$ has refuted several conjectures.

The most well known of them was suggested by Jean-Eric Pin in 1978. Pin conjectured that if a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with n states admits a word $w \in \Sigma^*$ such that $|Q \,.\, w| = k$, $1 \leq k \leq n$, then $\mathscr{A}$ possesses a word of length at most $(n-k)^2$ with the same property. (The Černý conjecture corresponds to the case $k = 1$.)

Kari's Example

Kari's automaton $\mathscr{K}_6$ has refuted several conjectures.

The most well known of them was suggested by Jean-Eric Pin in 1978. Pin conjectured that if a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with n states admits a word $w \in \Sigma^*$ such that $|Q \,.\, w| = k$, $1 \leq k \leq n$, then $\mathscr{A}$ possesses a word of length at most $(n-k)^2$ with the same property. (The Černý conjecture corresponds to the case $k = 1$.)

However, in $\mathscr{K}_6$ there is no word w of length $16 = (6-2)^2$ such that $|Q \,.\, w| = 2$.

Kari's Example

The *rank* of a DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is the minimum cardinality of the sets $\boldsymbol{Q} \,.\, \boldsymbol{w}$ where $\boldsymbol{w}$ runs over Σ^*.

Kari's Example

The *rank* of a DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is the minimum cardinality of the sets $\boldsymbol{Q} \,.\, \boldsymbol{w}$ where $\boldsymbol{w}$ runs over $\boldsymbol{\Sigma}^*$. This is the minimum score that can be achieved in the solitaire game on the automaton $\mathscr{A}$.

Kari's Example

The *rank* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \,.\, w$ where w runs over Σ^*. This is the minimum score that can be achieved in the solitaire game on the automaton $\mathscr{A}$. Synchronizing automata are precisely those of rank 1.

Kari's Example

The *rank* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \,.\, w$ where w runs over Σ^*. This is the minimum score that can be achieved in the solitaire game on the automaton $\mathscr{A}$. Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following rank conjecture:

Kari's Example

The *rank* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \,.\, w$ where w runs over Σ^*. This is the minimum score that can be achieved in the solitaire game on the automaton $\mathscr{A}$. Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following rank conjecture: if a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with n states has rank k, then there exists a word $w \in \Sigma^*$ of length at most $(n-k)^2$ such that $|Q \,.\, w| = k$.

Kari's Example

The *rank* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is the minimum cardinality of the sets $Q \mathbin{.} w$ where w runs over Σ^*. This is the minimum score that can be achieved in the solitaire game on the automaton $\mathscr{A}$. Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following rank conjecture: if a DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ with n states has rank k, then there exists a word $w \in \Sigma^*$ of length at most $(n-k)^2$ such that $|Q \mathbin{.} w| = k$.

Again, the Černý conjecture corresponds to the case $k = 1$.

Kari's Example

Kari's automaton does not refute the rank conjecture!

Kari's Example

Kari's automaton does not refute the rank conjecture!
In the solitaire game on $\mathscr{K}_{\mathbf{6}}$, no sequence of 16 moves removes 4 coins.

Kari's Example

Kari's automaton does not refute the rank conjecture! In the solitaire game on $\mathscr{K}_{\mathbf{6}}$, no sequence of 16 moves removes 4 coins. However, 4 is not the maximum number of tokens that can be removed!

Kari's Example

Kari's automaton does not refute the rank conjecture! In the solitaire game on $\mathscr{K}_{\mathbf{6}}$, no sequence of 16 moves removes 4 coins. However, 4 is not the maximum number of tokens that can be removed! One can show that 5 states can be freed by a sequence of 25 moves — in full accordance with the rank conjecture.

Kari's Example

Kari's automaton does not refute the rank conjecture! In the solitaire game on $\mathscr{K}_{\mathbf{6}}$, no sequence of 16 moves removes 4 coins. However, 4 is not the maximum number of tokens that can be removed! One can show that 5 states can be freed by a sequence of 25 moves — in full accordance with the rank conjecture.

Yet another hope killed by Kari's example is the extensibility conjecture.

Kari's Example

Kari's automaton does not refute the rank conjecture! In the solitaire game on $\mathscr{K}_{\mathbf{6}}$, no sequence of 16 moves removes 4 coins. However, 4 is not the maximum number of tokens that can be removed! One can show that 5 states can be freed by a sequence of 25 moves — in full accordance with the rank conjecture.

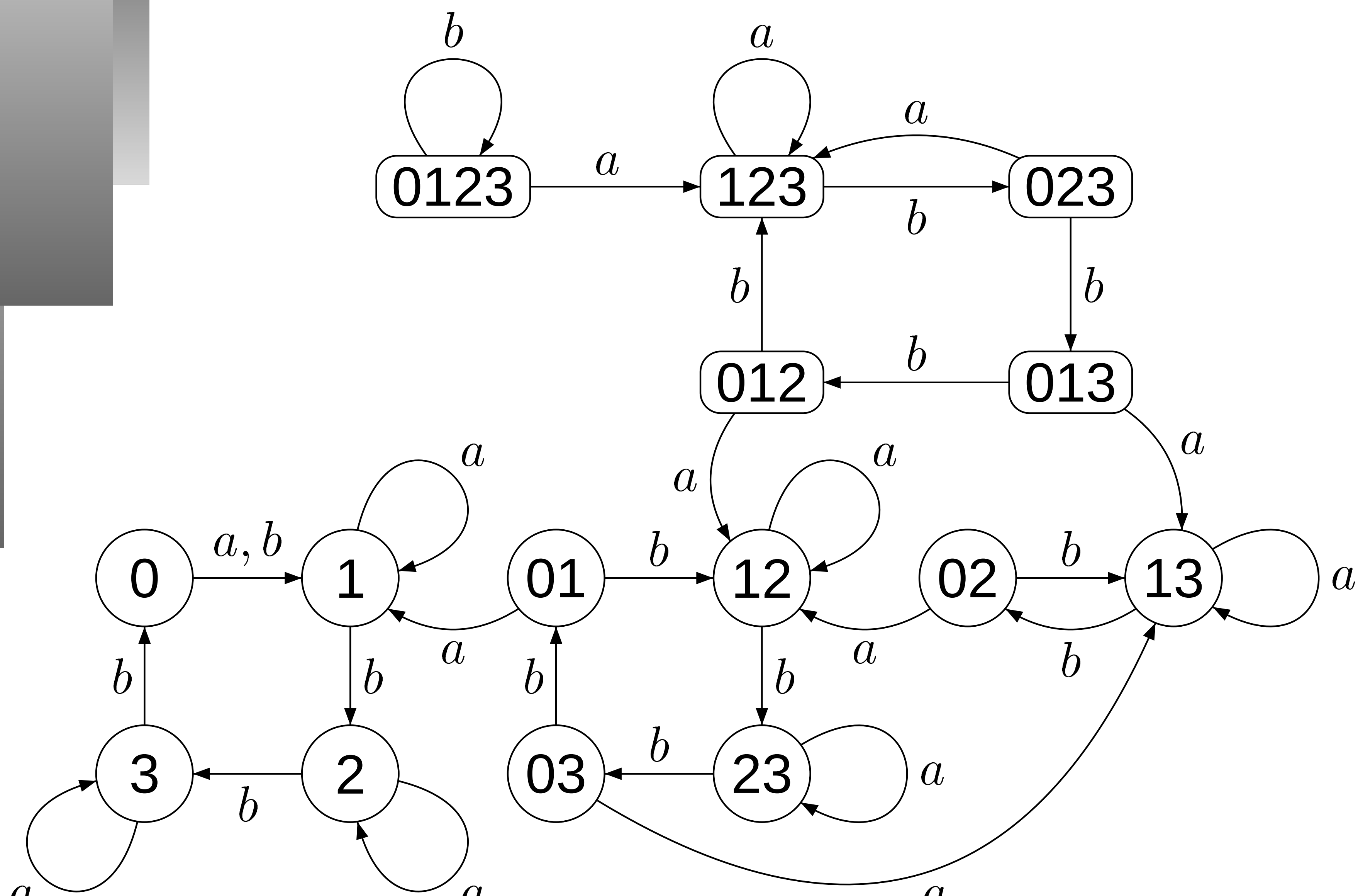
Yet another hope killed by Kari's example is the extensibility conjecture. In a DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$, a subset $\boldsymbol{P} \subset \boldsymbol{Q}$ is *extensible* if $\boldsymbol{P} = \boldsymbol{R} \,.\, \boldsymbol{w}$ for some $\boldsymbol{w} \in \boldsymbol{\Sigma}^*$ of length at most $\boldsymbol{n} = |\boldsymbol{Q}|$ and some $\boldsymbol{R} \subseteq \boldsymbol{Q}$ with $|\boldsymbol{R}| > |\boldsymbol{P}|$.

Kari’s Example

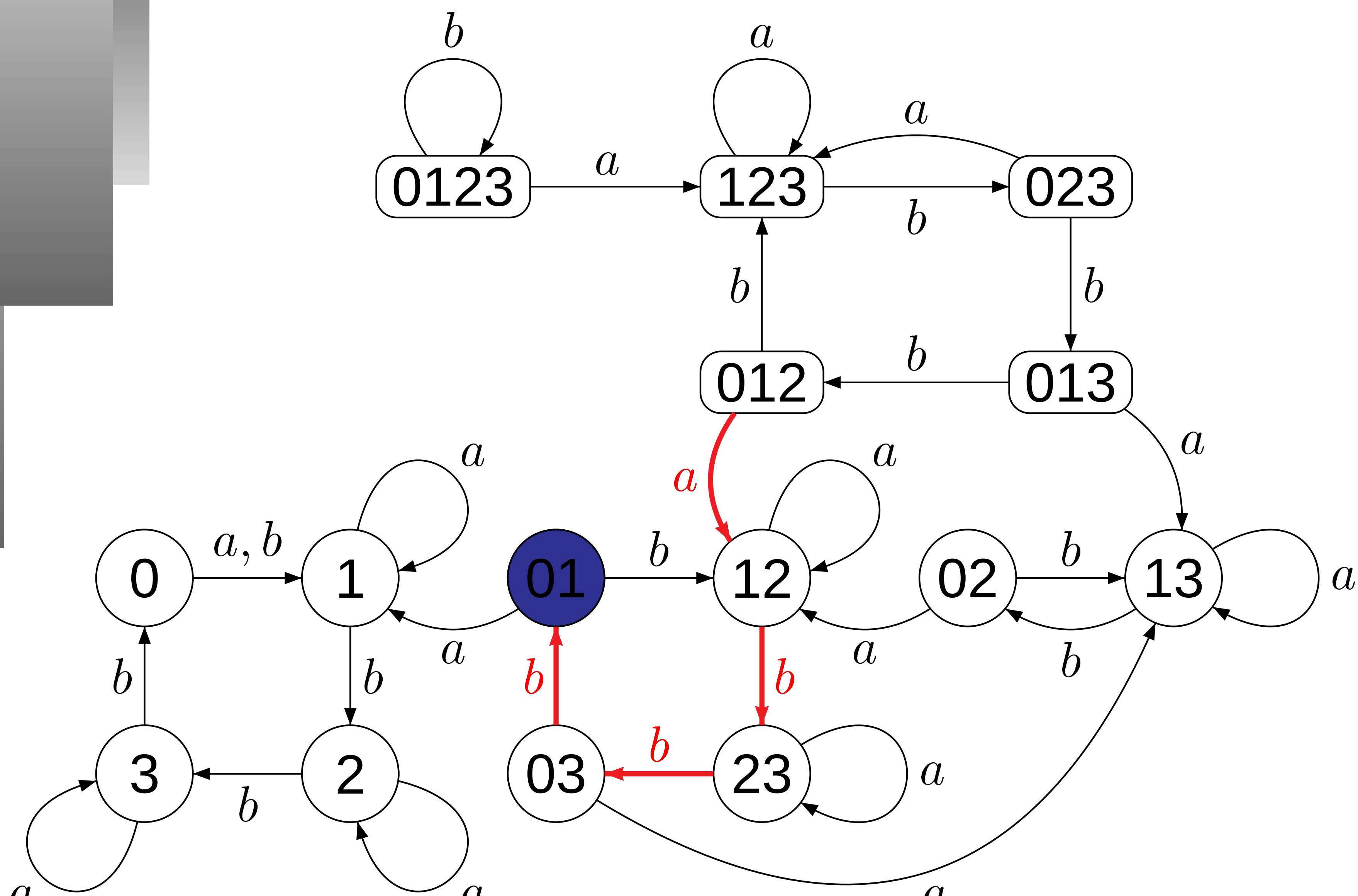
Kari’s automaton does not refute the rank conjecture! In the solitaire game on $\mathscr{K}_{\mathbf{6}}$, no sequence of 16 moves removes 4 coins. However, 4 is not the maximum number of tokens that can be removed! One can show that 5 states can be freed by a sequence of 25 moves — in full accordance with the rank conjecture.

Yet another hope killed by Kari’s example is the extensibility conjecture. In a DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$, a subset $\boldsymbol{P} \subset \boldsymbol{Q}$ is *extensible* if $\boldsymbol{P} = \boldsymbol{R} \, . \, \boldsymbol{w}$ for some $\boldsymbol{w} \in \Sigma^*$ of length at most $\boldsymbol{n} = |\boldsymbol{Q}|$ and some $\boldsymbol{R} \subseteq \boldsymbol{Q}$ with $|\boldsymbol{R}| > |\boldsymbol{P}|$. It was conjectured that in synchronizing automata every proper non-singleton subset is extensible.

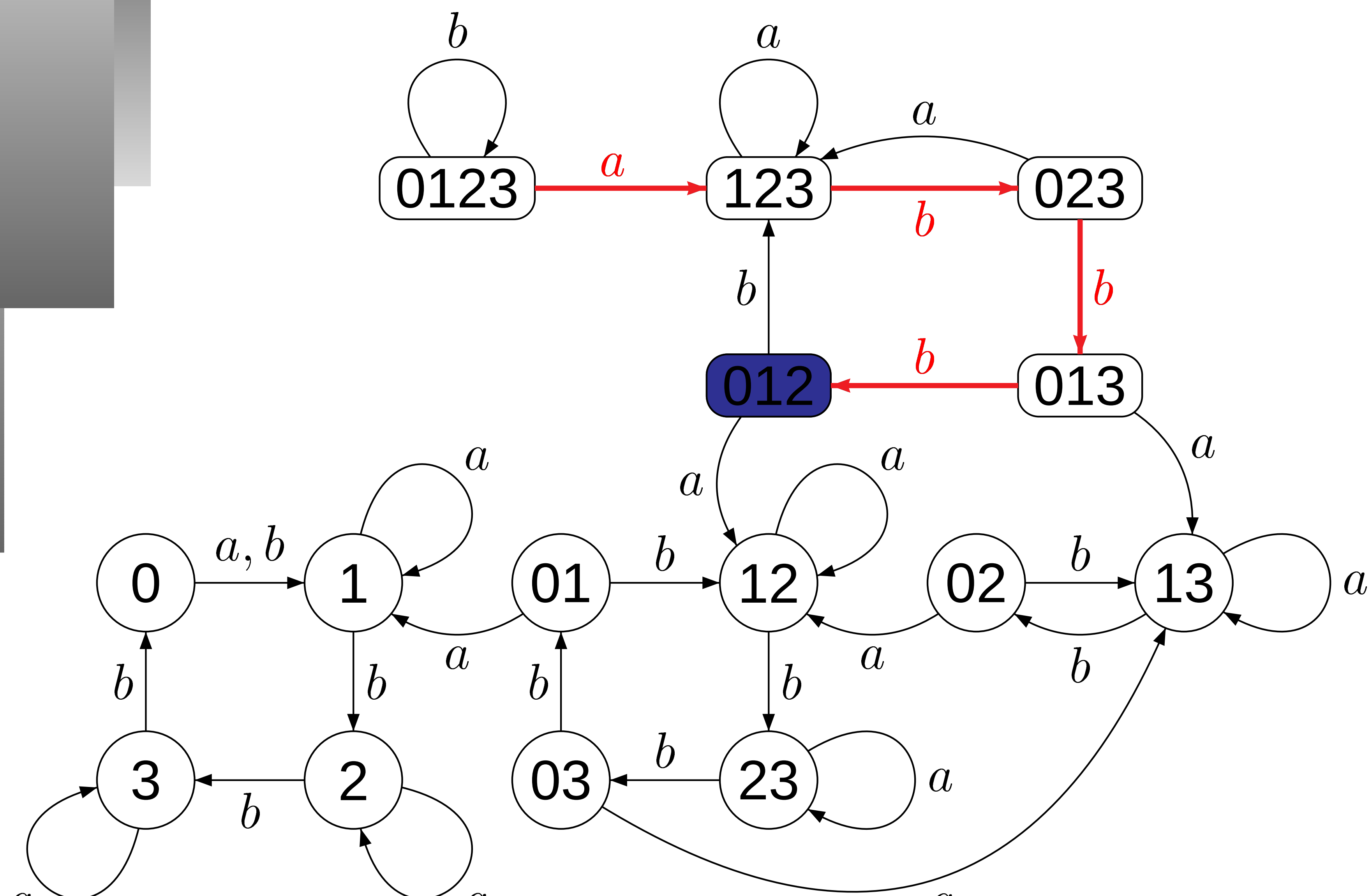
Extensibility



Extensibility



Extensibility



Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.

Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should sent two states $q, q' \in Q$ to the same state p.

Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.
Indeed, if $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should sent two states $q, q' \in Q$ to the same state p. Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \mathbin{.} w_i$ for some word w_i of length $\leq n$.

Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should sent two states $q, q' \in Q$ to the same state p. Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \,.\, w_i$ for some word w_i of length $\leq n$. Then in at most $n - 2$ steps the sequence $P_0, P_1, P_2, \ldots$ reaches Q and

$$Q \,.\, w_{n-2} w_{n-1} \cdots w_1 a = \{p\},$$

that is, $w_{n-2} w_{n-1} \cdots w_1 a$ is a reset word.

Extensibility

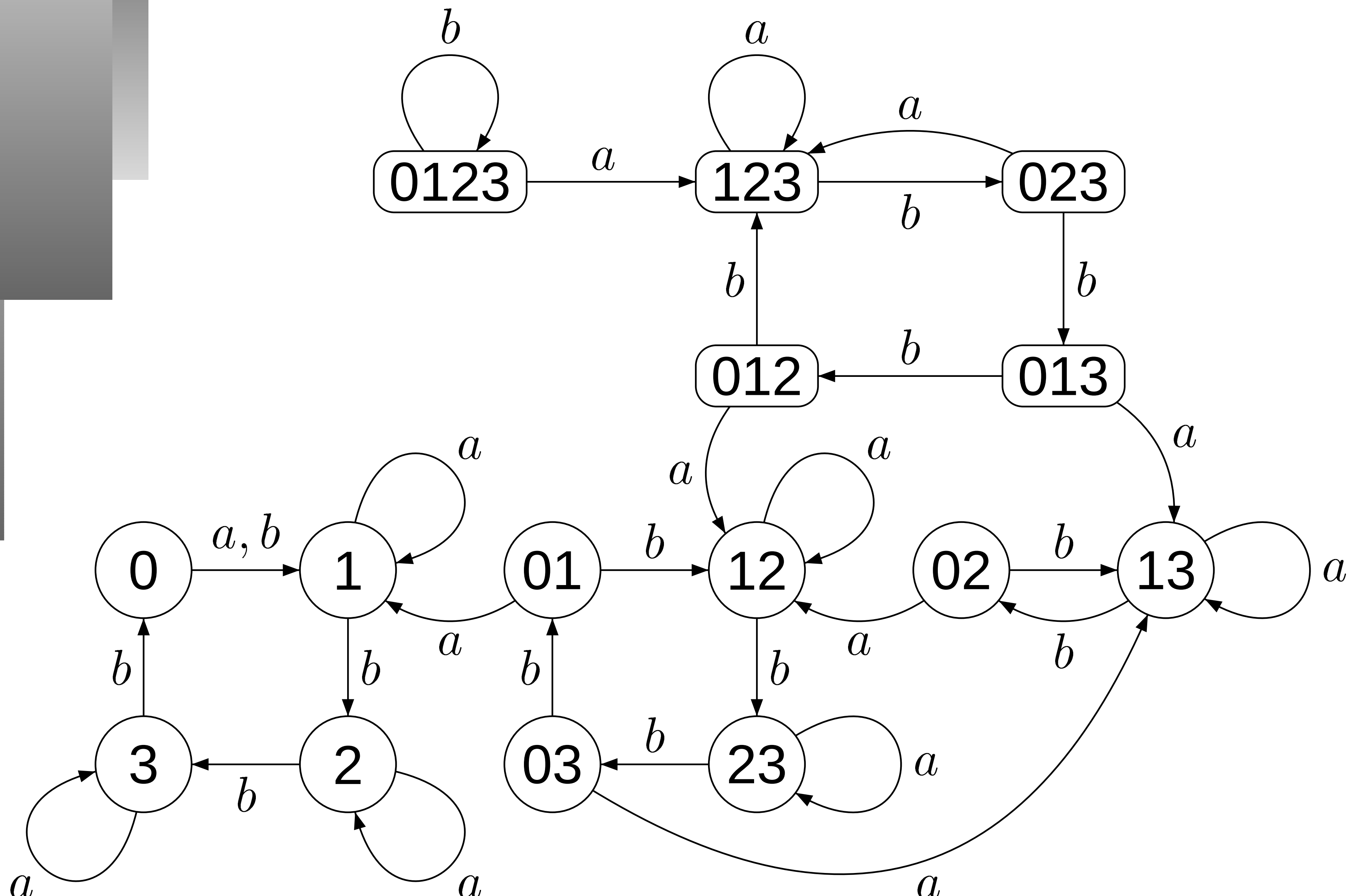
Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should sent two states $q, q' \in Q$ to the same state p. Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \mathbin{.} w_i$ for some word w_i of length $\leq n$. Then in at most $n-2$ steps the sequence $P_0, P_1, P_2, \dots$ reaches Q and

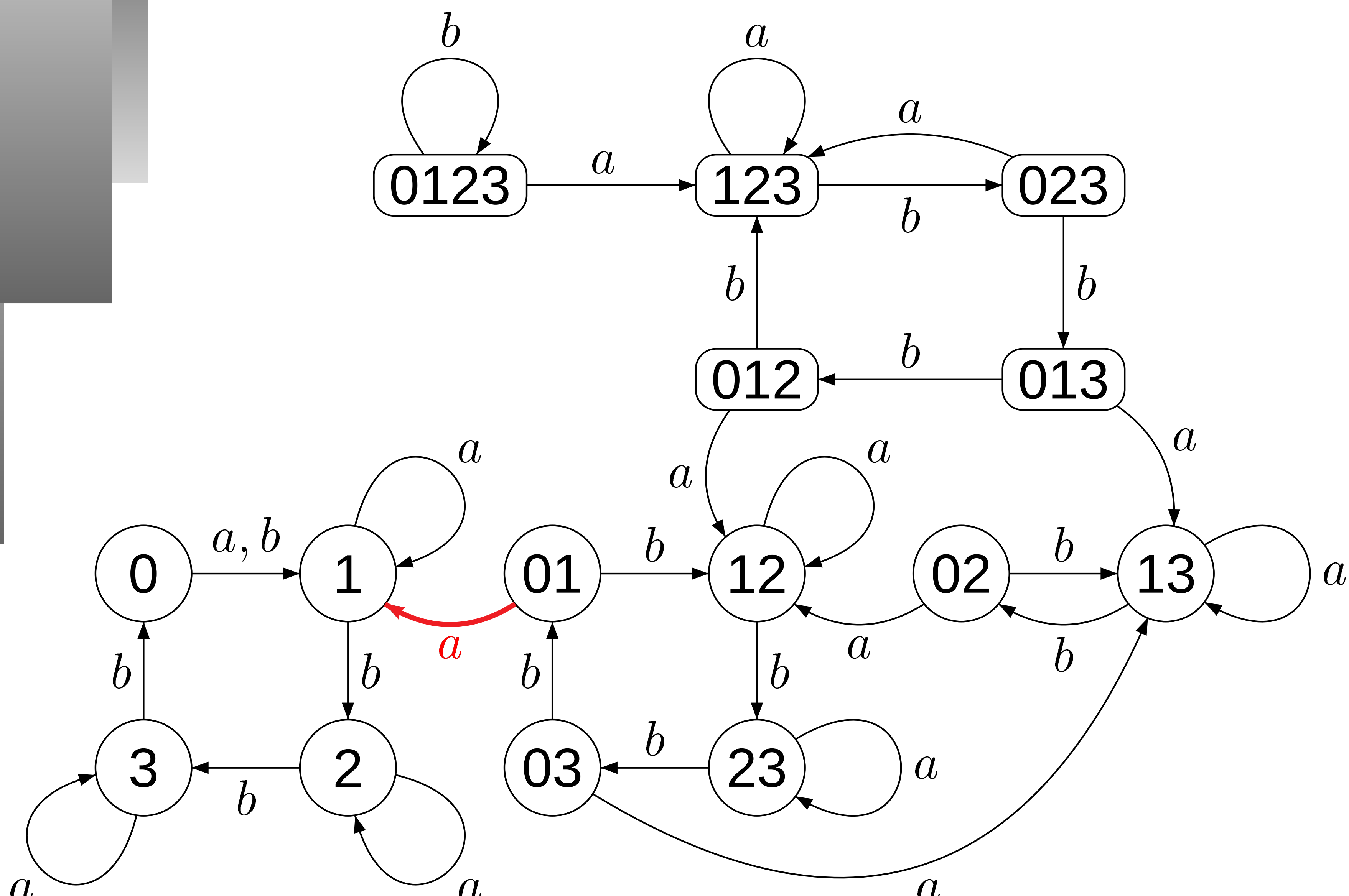
$$Q \mathbin{.} w_{n-2} w_{n-1} \cdots w_1 a = \{p\},$$

that is, $w_{n-2} w_{n-1} \cdots w_1 a$ is a reset word. The length of this reset word is at most $n(n-2)+1 = (n-1)^2$.

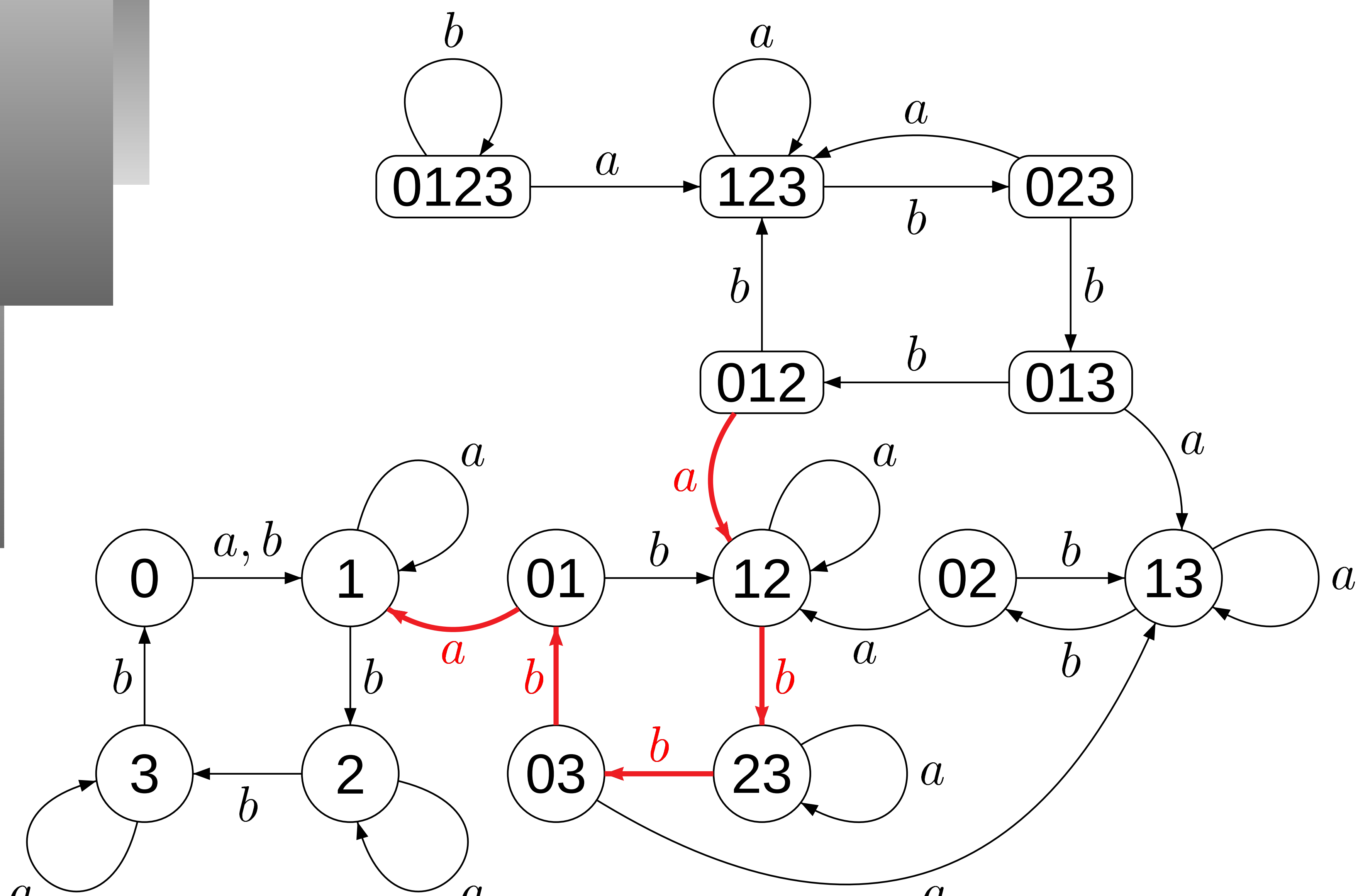
Extensibility



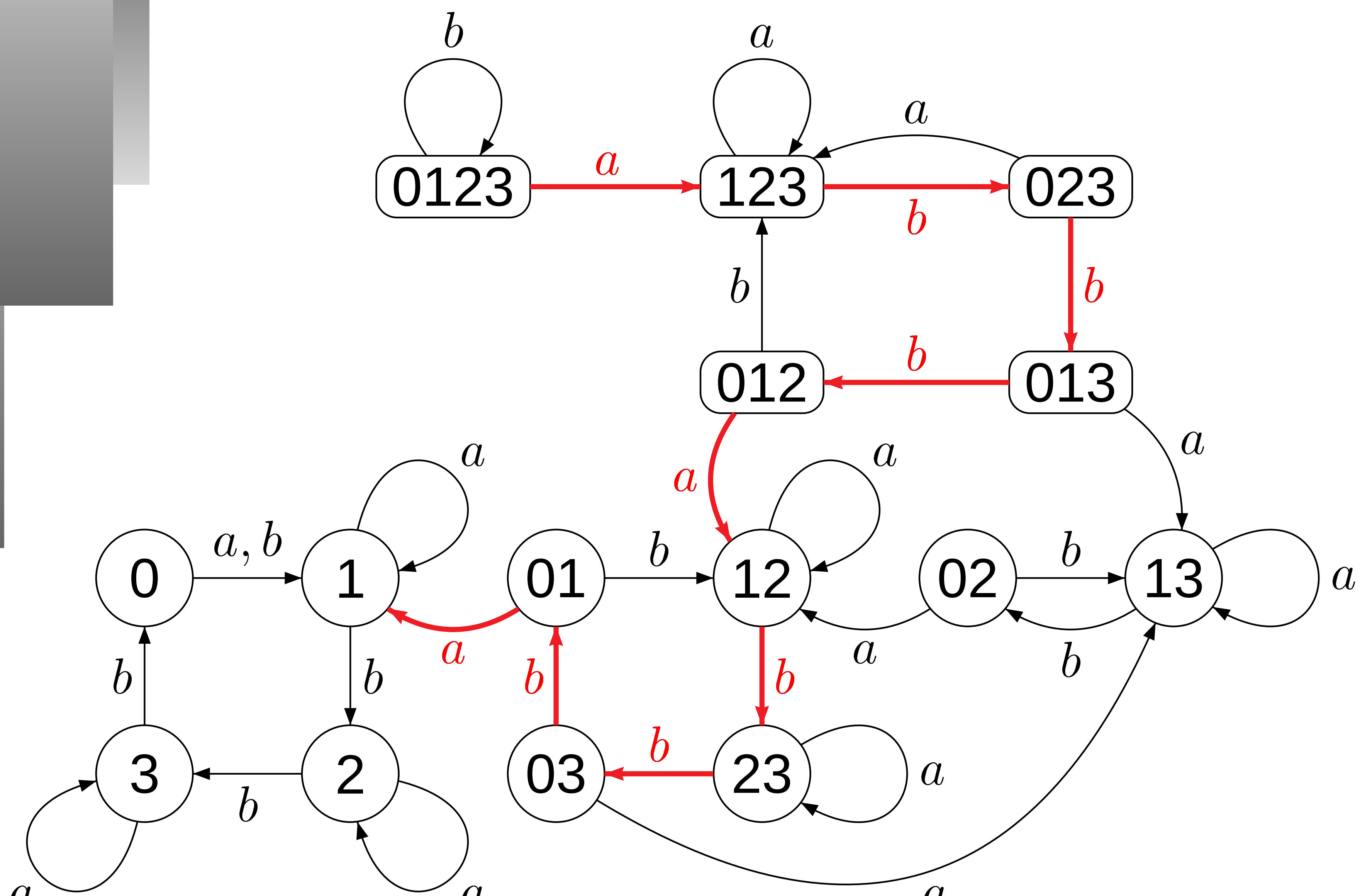
Extensibility



Extensibility



Extensibility



Extensibility

Several important results confirming the Černý conjecture for various partial cases have been proved by verifying the extensibility conjecture for the corresponding automata. This includes, in particular:

Extensibility

Several important results confirming the Černý conjecture for various partial cases have been proved by verifying the extensibility conjecture for the corresponding automata. This includes, in particular:

- Louis Dubuc's result for automata in which a letter acts on the state set Q as a cyclic permutation of order $|Q|$ (Sur le automates circulaires et la conjecture de Černý, RAIRO Inform. Theor. Appl., 32 (1998) 21–34 [in French]).

Extensibility

Several important results confirming the Černý conjecture for various partial cases have been proved by verifying the extensibility conjecture for the corresponding automata. This includes, in particular:

• Louis Dubuc's result for automata in which a letter acts on the state set Q as a cyclic permutation of order $|Q|$ (Sur le automates circulaires et la conjecture de Černý, RAIRO Inform. Theor. Appl., 32 (1998) 21–34 [in French]).

• Jarkko Kari's result for automata with Eulerian digraphs (Synchronizing finite automata on Eulerian digraphs, Theoret. Comput. Sci., 295 (2003) 223–232.)

Extensibility vs Kari's Example

However, in $\mathscr{K}_{\mathbf{6}}$ there exists a 2-subset that cannot be extended to a larger subset by any word of length 6 (and even by any word of length 7).

Extensibility vs Kari’s Example

However, in $\mathscr{K}_{\mathbf{6}}$ there exists a 2-subset that cannot be extended to a larger subset by any word of length 6 (and even by any word of length 7).
Thus, the extensibility conjecture fails, and the approach based on it cannot prove the Černý conjecture in general.

Strongly connected digraphs

Studying synchronizing automata, it is natural to restrict to the strongly connected case.

Strongly connected digraphs

Studying synchronizing automata, it is natural to restrict to the strongly connected case. For instance, it suffices to prove the Černý conjecture for this case, the general case would be an easy consequence.

Strongly connected digraphs

Studying synchronizing automata, it is natural to restrict to the strongly connected case. For instance, it suffices to prove the Černý conjecture for this case, the general case would be an easy consequence.

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a synchronizing automaton with n states.

Strongly connected digraphs

Studying synchronizing automata, it is natural to restrict to the strongly connected case. For instance, it suffices to prove the Černý conjecture for this case, the general case would be an easy consequence.

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a synchronizing automaton with n states. Consider the set S of all states to which $\mathscr{A}$ can be synchronized and let $m = |S|$. If $q \in S$, then there exists a reset word $w \in \Sigma^*$ such that $Q.w = \{q\}$. For each $a \in \Sigma$, we have $Q.wa = \{\delta(q, a)\}$ whence wa also is a reset word and $\delta(q, a) \in S$. Thus, restricting the function δ to $S \times \Sigma$, we get a subautomaton $\mathscr{S}$ with the state set S.

Strongly connected digraphs

Studying synchronizing automata, it is natural to restrict to the strongly connected case. For instance, it suffices to prove the Černý conjecture for this case, the general case would be an easy consequence.

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a synchronizing automaton with n states. Consider the set S of all states to which $\mathscr{A}$ can be synchronized and let $m = |S|$. If $q \in S$, then there exists a reset word $w \in \Sigma^*$ such that $Q.w = \{q\}$. For each $a \in \Sigma$, we have $Q.wa = \{\delta(q, a)\}$ whence wa also is a reset word and $\delta(q, a) \in S$. Thus, restricting the function δ to $S \times \Sigma$, we get a subautomaton $\mathscr{S}$ with the state set S. Obviously, $\mathscr{S}$ is synchronizing and strongly connected.

Strongly connected digraphs

If the Černý conjecture holds true for strongly connected synchronizing automata, $\mathscr{S}$ has a reset word v of length $(m-1)^2$.

Strongly connected digraphs

If the Černý conjecture holds true for strongly connected synchronizing automata, $\mathscr{S}$ has a reset word v of length $(m-1)^2$.

Now consider the partition π of Q into $n-m+1$ classes one of which is S and all others are singletons. Then π is a congruence of the automaton $\mathscr{A}$.

Strongly connected digraphs

If the Černý conjecture holds true for strongly connected synchronizing automata, $\mathscr{S}$ has a reset word v of length $(m-1)^2$.

Now consider the partition π of Q into $n-m+1$ classes one of which is S and all others are singletons. Then π is a congruence of the automaton $\mathscr{A}$.

We recall the notion of a congruence and the related notion of the quotient automaton w.r.t. a congruence in the next slide. They will be essentially used in this lecture!

Congruences and Quotient Automata

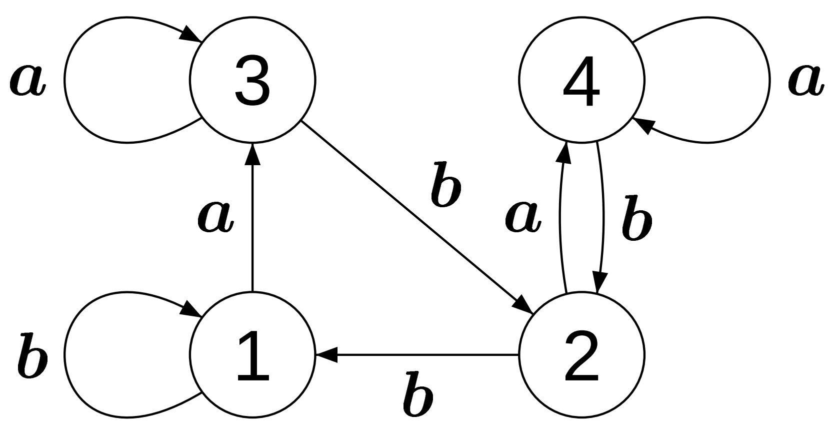
An equivalence π on the state set Q of a DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is called a *congruence* if $(\boldsymbol{p}, \boldsymbol{q}) \in \boldsymbol{\pi}$ implies $\big(\boldsymbol{\delta}(\boldsymbol{p}, \boldsymbol{a}), \boldsymbol{\delta}(\boldsymbol{q}, \boldsymbol{a})\big) \in \boldsymbol{\pi}$ for all $\boldsymbol{p}, \boldsymbol{q} \in \boldsymbol{Q}$ and all $\boldsymbol{a} \in \boldsymbol{\Sigma}$.

Congruences and Quotient Automata

An equivalence π on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called a *congruence* if $(p, q) \in \pi$ implies $\big(\delta(p, a), \delta(q, a)\big) \in \pi$ for all $p, q \in Q$ and all $a \in \Sigma$. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.

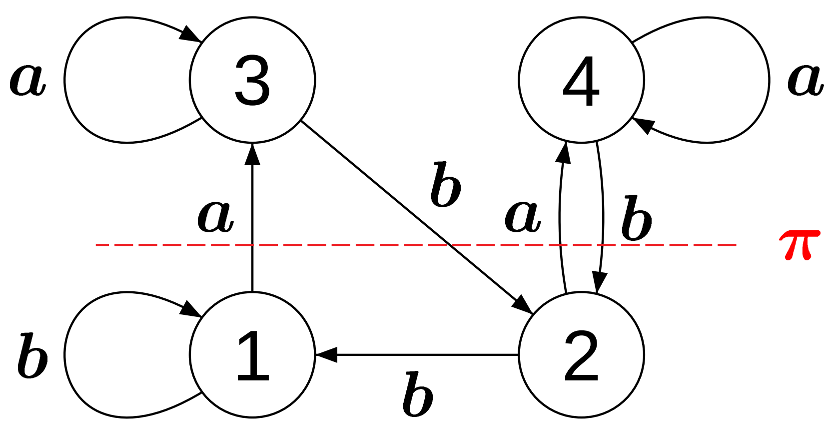
Congruences and Quotient Automata

An equivalence π on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called a *congruence* if $(p, q) \in \pi$ implies $\big(\delta(p, a), \delta(q, a)\big) \in \pi$ for all $p, q \in Q$ and all $a \in \Sigma$. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.



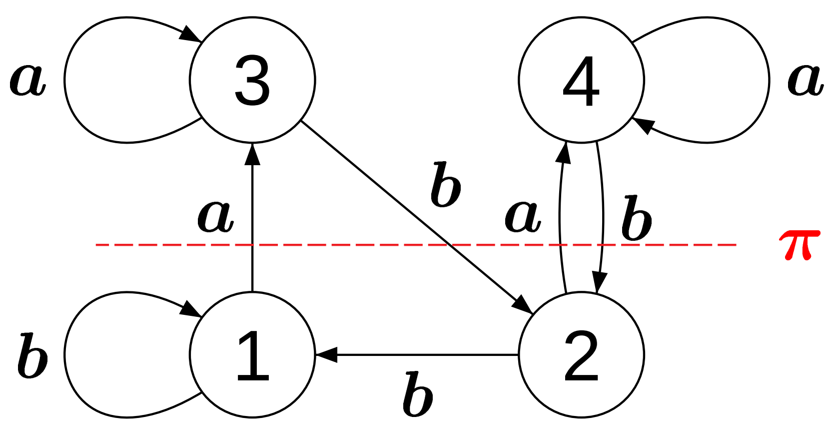
Congruences and Quotient Automata

An equivalence π on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called a *congruence* if $(p, q) \in \pi$ implies $\big(\delta(p, a), \delta(q, a)\big) \in \pi$ for all $p, q \in Q$ and all $a \in \Sigma$. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.



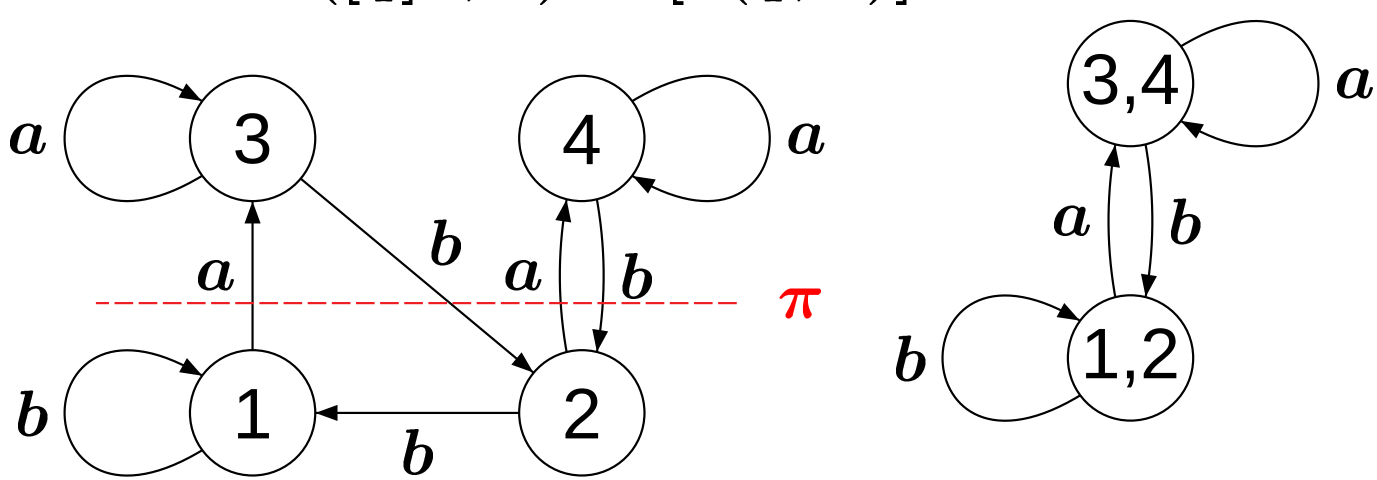
Congruences and Quotient Automata

An equivalence π on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called a *congruence* if $(p, q) \in \pi$ implies $\big(\delta(p,a), \delta(q,a)\big) \in \pi$ for all $p, q \in Q$ and all $a \in \Sigma$. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.
The *quotient* $\mathscr{A}/\pi$ is the DFA $\langle Q/\pi, \Sigma, \delta_\pi \rangle$ where $Q/\pi = \{[q]_\pi \mid q \in Q\}$ and the function δ_π is defined by the rule $\delta_\pi([q]_\pi, a) = [\delta(q,a)]_\pi$.



Congruences and Quotient Automata

An equivalence π on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called a *congruence* if $(p, q) \in \pi$ implies $\big(\delta(p, a), \delta(q, a)\big) \in \pi$ for all $p, q \in Q$ and all $a \in \Sigma$. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.
The *quotient* $\mathscr{A}/\pi$ is the DFA $\langle Q/\pi, \Sigma, \delta_\pi \rangle$ where $Q/\pi = \{[q]_\pi \mid q \in Q\}$ and the function δ_π is defined by the rule $\delta_\pi([q]_\pi, a) = [\delta(q, a)]_\pi$.



Strongly connected digraphs

Return to our reasoning: let π be the partition of Q into $n - m + 1$ classes one of which is S and all others are singletons. Then π is a congruence of $\mathscr{A}$.

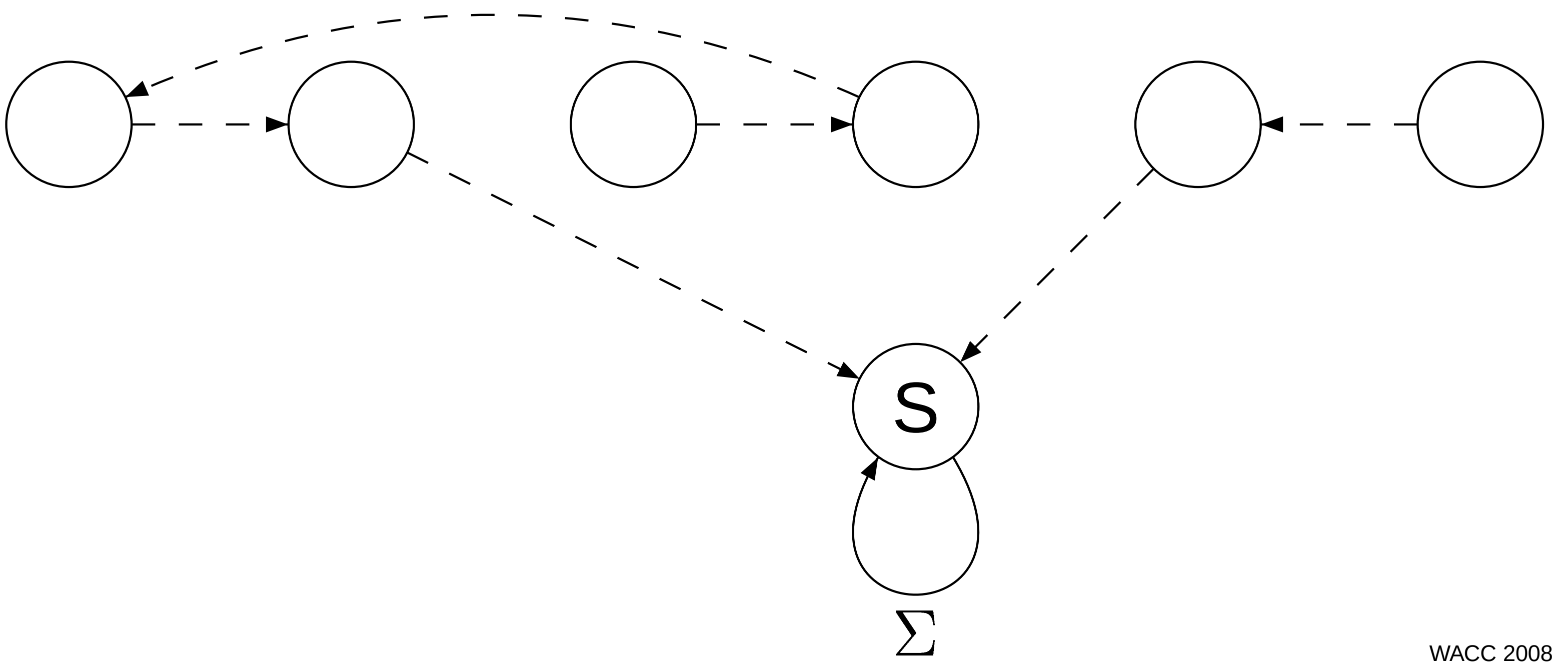
Strongly connected digraphs

Return to our reasoning: let π be the partition of Q into $n - m + 1$ classes one of which is S and all others are singletons. Then π is a congruence of $\mathscr{A}$.

Clearly, the quotient $\mathscr{A}/\pi$ is synchronizing and has S as a unique sink.

Strongly connected digraphs

Return to our reasoning: let π be the partition of Q into $n - m + 1$ classes one of which is S and all others are singletons. Then π is a congruence of $\mathscr{A}$.
Clearly, the quotient $\mathscr{A}/\pi$ is synchronizing and has S as a unique sink.

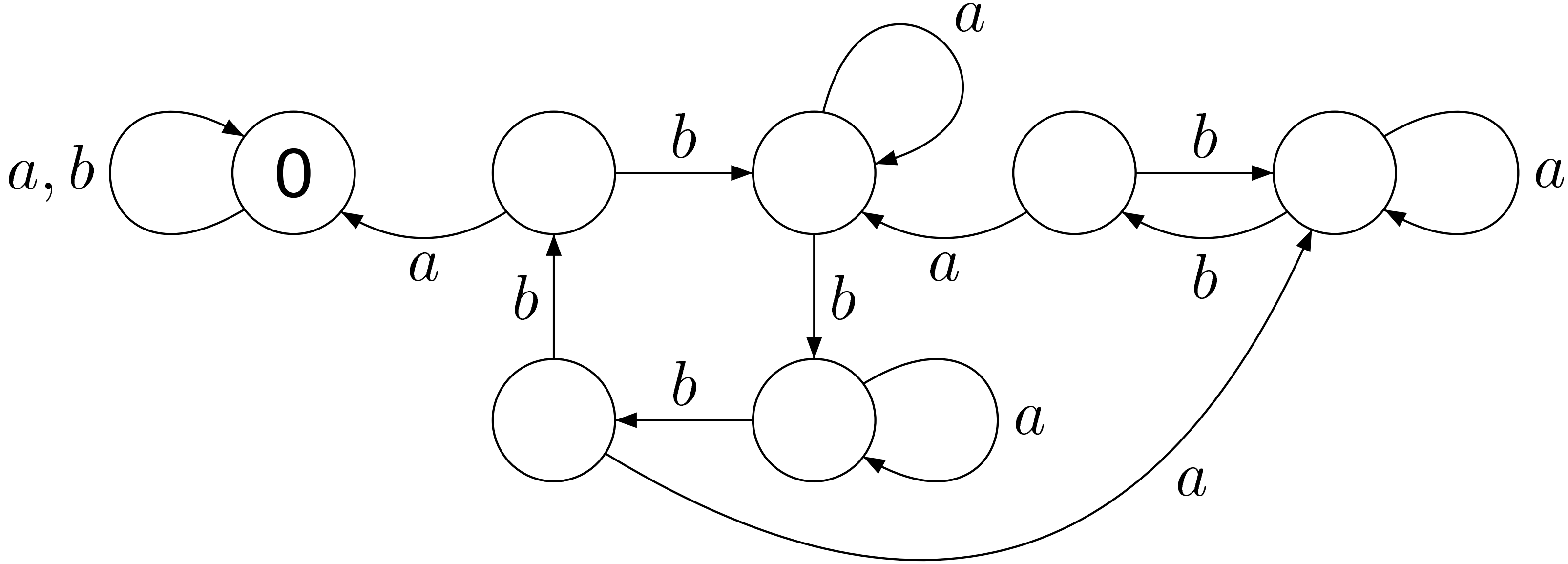


Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.

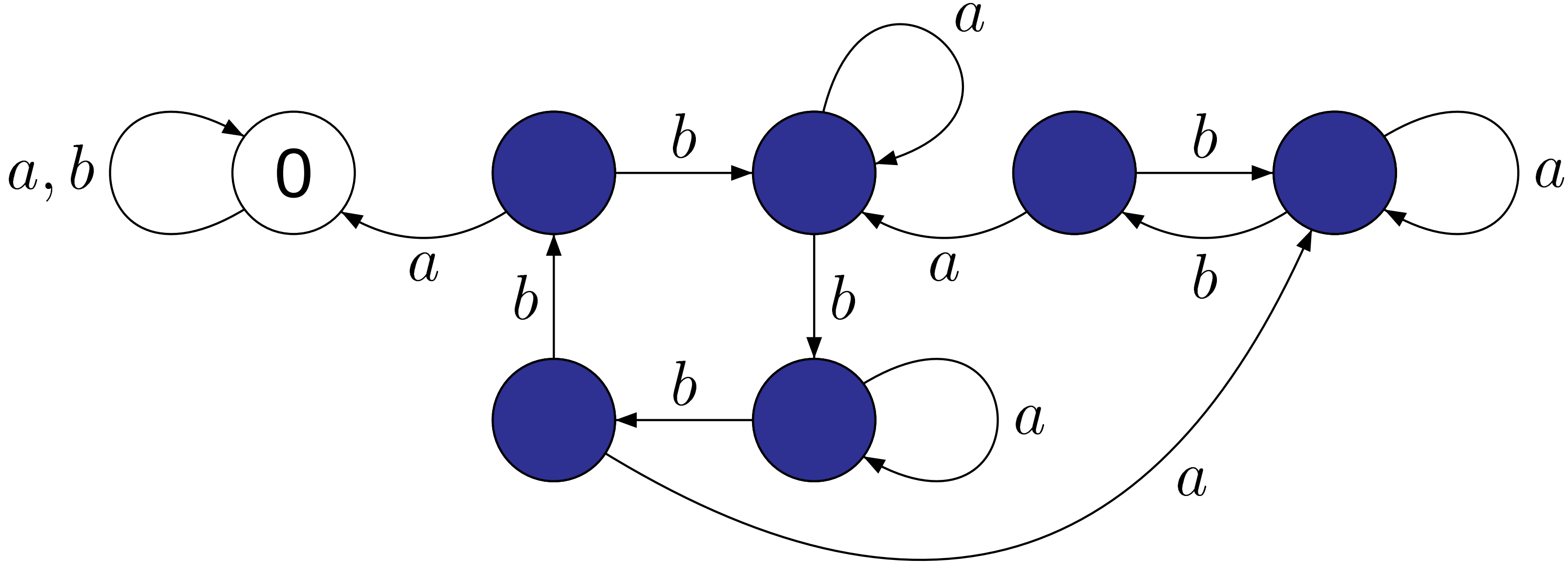
Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



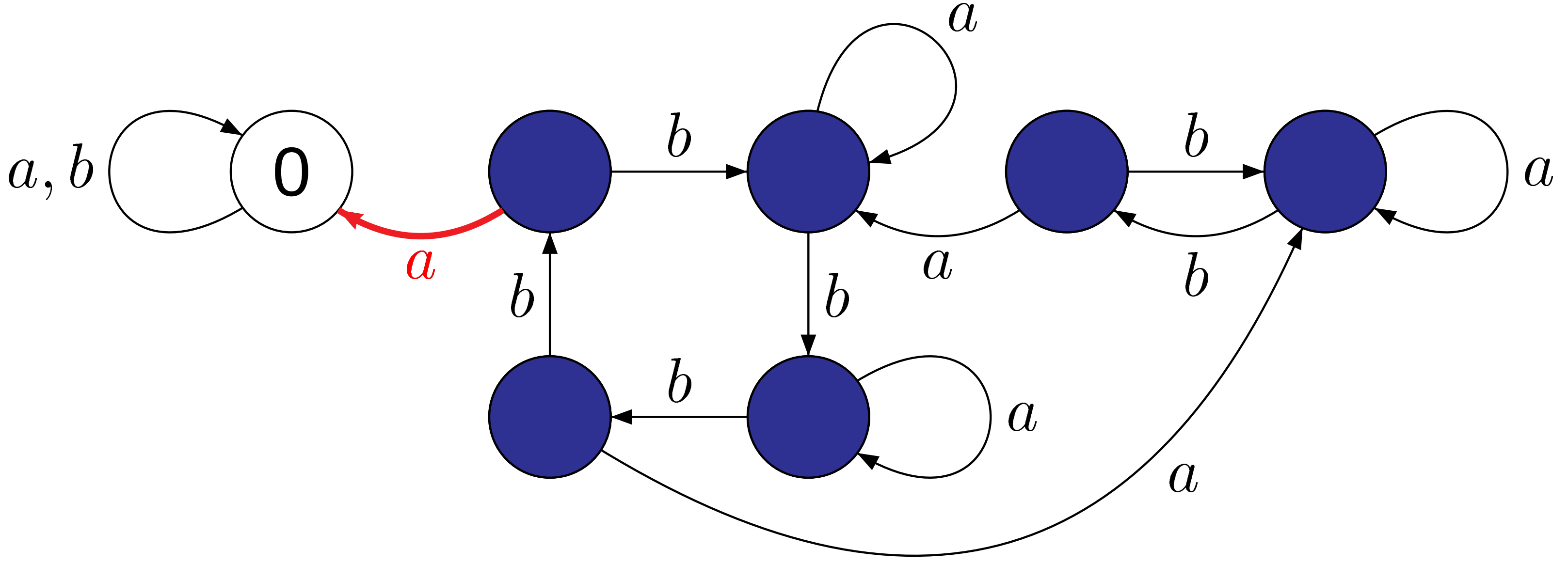
Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



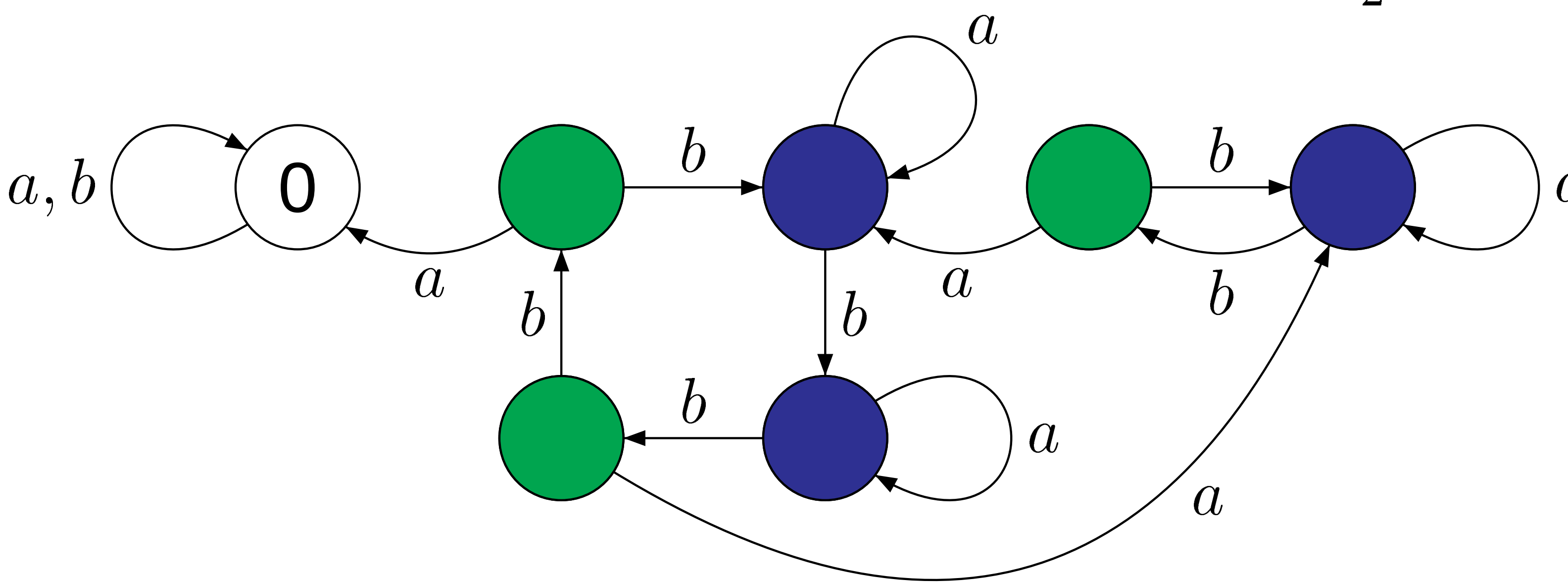
Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



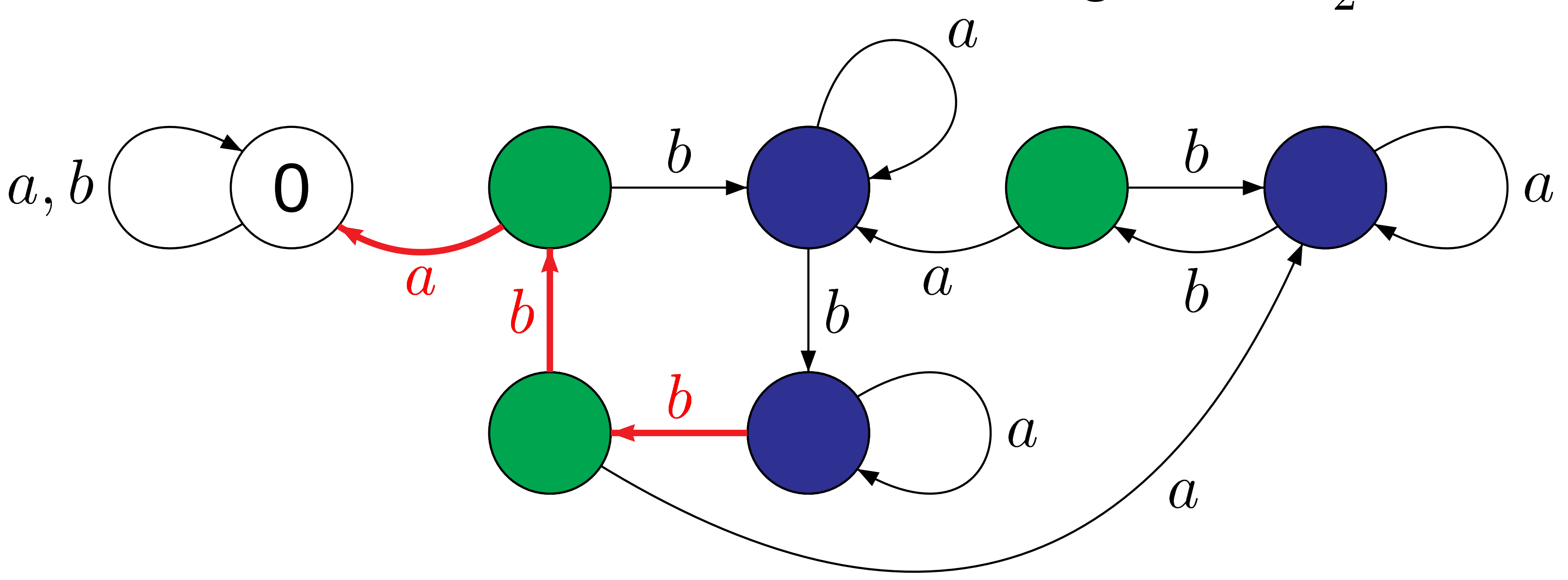
Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



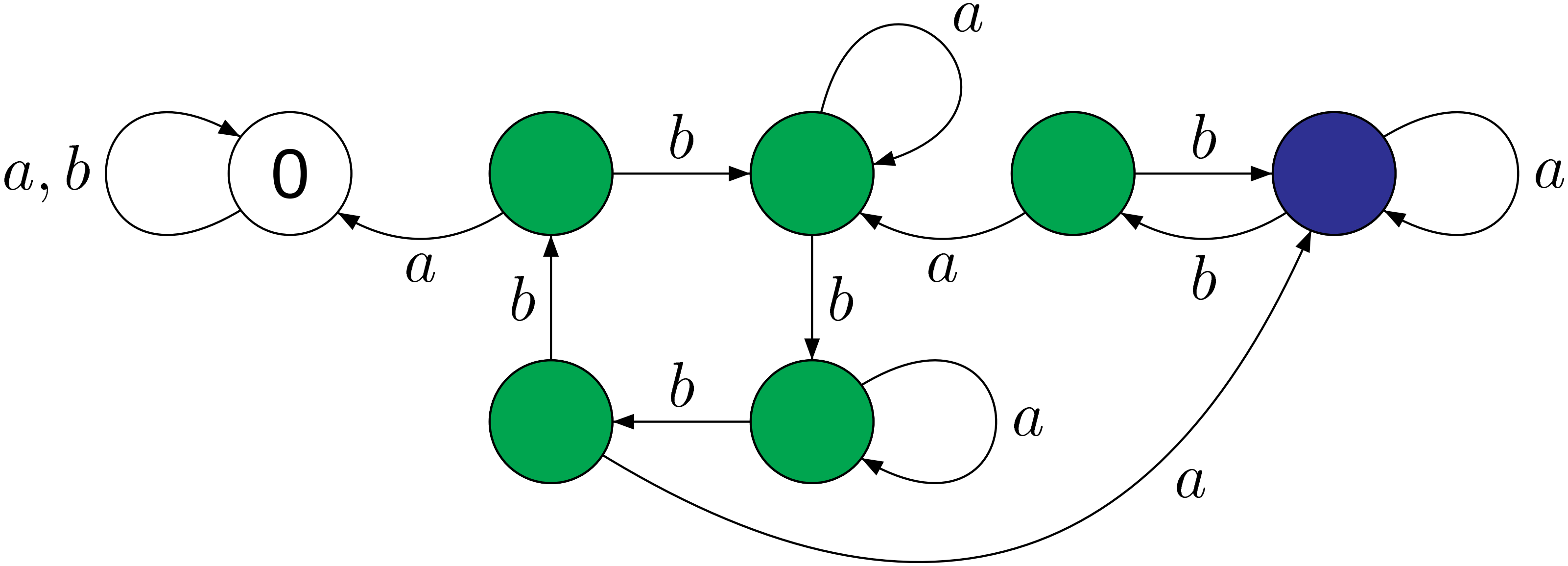
Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



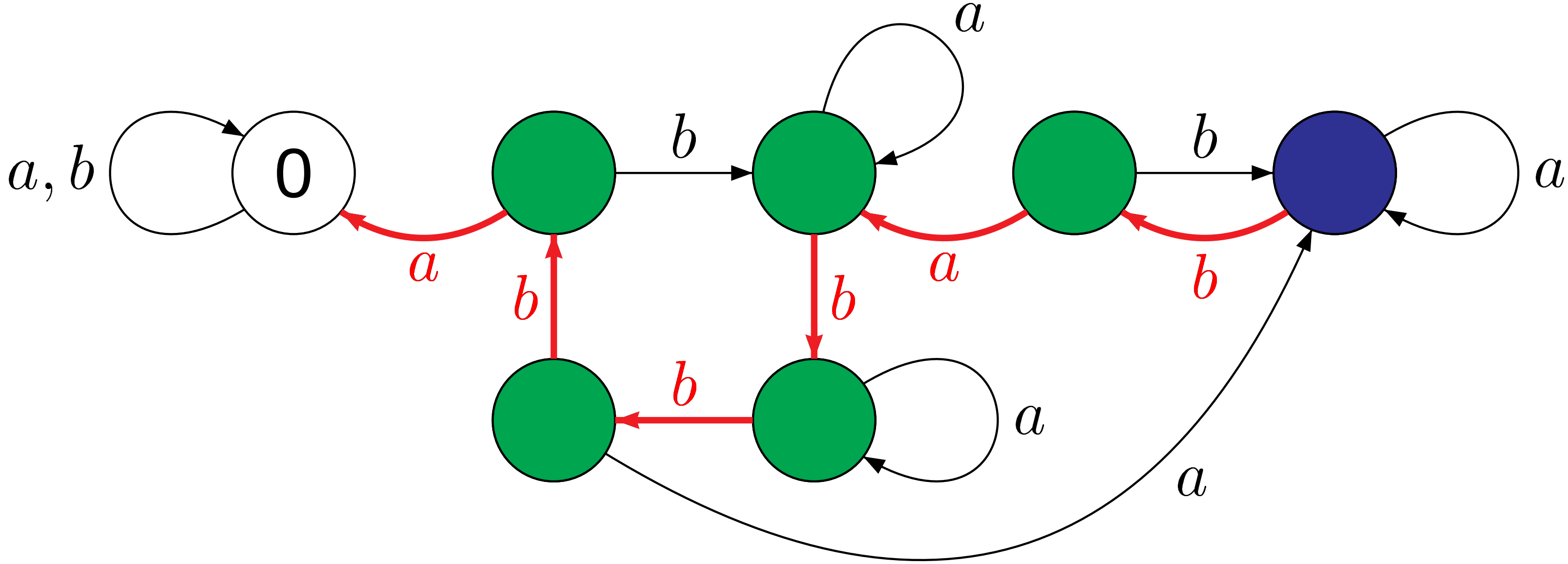
Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



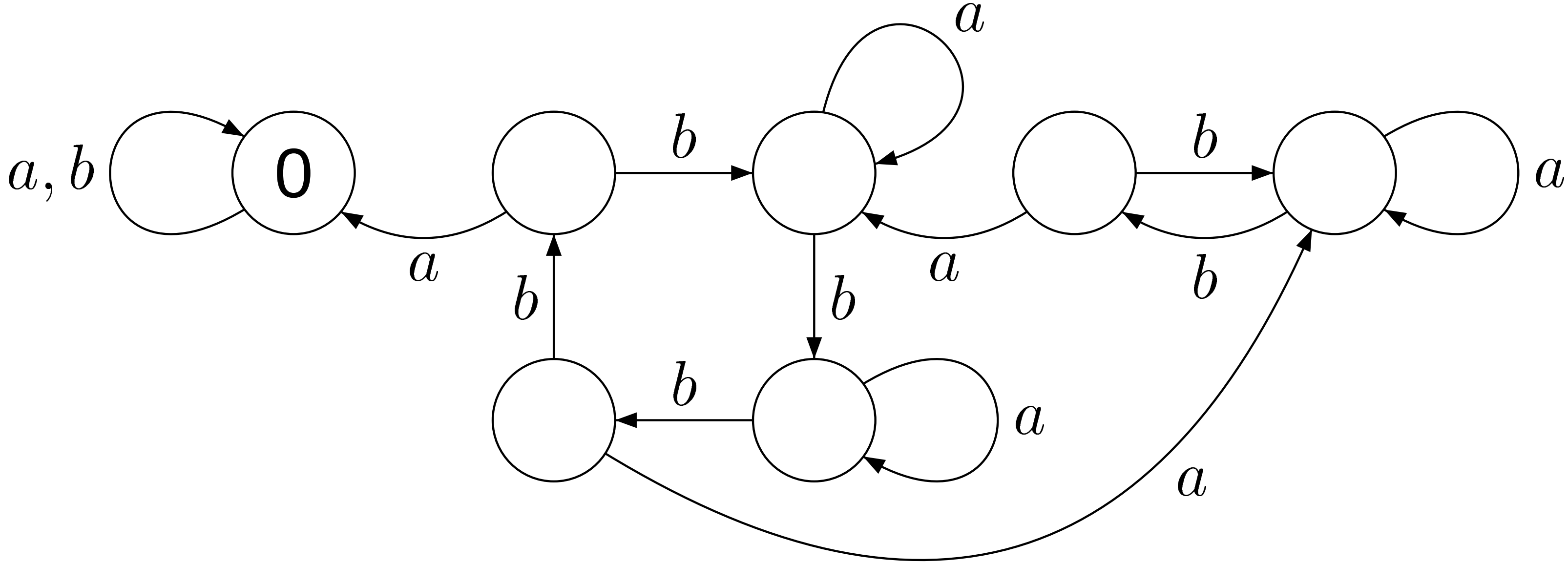
Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



Automata with a Unique Sink

If a synchronizing automata with k states has a unique sink, then it has a reset word of length $\leq \frac{k(k-1)}{2}$.



The algorithm makes at most $k-1$ steps and the length of the segment added in the step when t states still holds coins ($k-1 \geq t \geq 1$) is at most $k-t$. The total length is $\leq 1+2+\cdots+(k-1) = \frac{k(k-1)}{2}$.

Strongly connected digraphs

Return to our reasoning: the quotient $\mathscr{A}/\pi$ is synchronizing with a unique sink and has $n - m + 1$ states.

Strongly connected digraphs

Return to our reasoning: the quotient $\mathscr{A}/\pi$ is synchronizing with a unique sink and has $n-m+1$ states. Hence, $\mathscr{A}/\pi$ has a reset word u of length $\frac{(n-m+1)(n-m)}{2}$.

Strongly connected digraphs

Return to our reasoning: the quotient $\mathscr{A}/\pi$ is synchronizing with a unique sink and has $n-m+1$ states. Hence, $\mathscr{A}/\pi$ has a reset word u of length $\frac{(n-m+1)(n-m)}{2}$. Then $Q\,.\,u \subseteq S$.

Strongly connected digraphs

Return to our reasoning: the quotient $\mathscr{A}/\pi$ is synchronizing with a unique sink and has $n-m+1$ states. Hence, $\mathscr{A}/\pi$ has a reset word u of length $\frac{(n-m+1)(n-m)}{2}$. Then $Q\,.\,u \subseteq S$.
Recall that we have assumed that the automaton $\mathscr{S}$ has a reset word v of length $(m-1)^2$.

Strongly connected digraphs

Return to our reasoning: the quotient $\mathscr{A}/\pi$ is synchronizing with a unique sink and has $n - m + 1$ states. Hence, $\mathscr{A}/\pi$ has a reset word u of length $\frac{(n-m+1)(n-m)}{2}$. Then $Q\,.\,u \subseteq S$.

Recall that we have assumed that the automaton $\mathscr{S}$ has a reset word v of length $(m-1)^2$. Then $S.v$ is a singleton, whence also $Q.uv \subseteq S.v$ is a singleton.

Strongly connected digraphs

Return to our reasoning: the quotient $\mathscr{A}/\pi$ is synchronizing with a unique sink and has $n-m+1$ states. Hence, $\mathscr{A}/\pi$ has a reset word u of length $\frac{(n-m+1)(n-m)}{2}$. Then $Q\,.\,u \subseteq S$.
Recall that we have assumed that the automaton $\mathscr{S}$ has a reset word v of length $(m-1)^2$. Then $S.v$ is a singleton, whence also $Q.uv \subseteq S.v$ is a singleton. Thus, uv is reset word for $\mathscr{A}$, and the length of this word does not exceed

$$\frac{(n-m+1)(n-m)}{2} + (m-1)^2 \leq (n-1)^2.$$

Strongly connected digraphs

Thus, we assume that our synchronizing automata are strongly connected as digraphs.

Strongly connected digraphs

Thus, we assume that our synchronizing automata are strongly connected as digraphs.
Observe that such an automaton can be reset to any state.

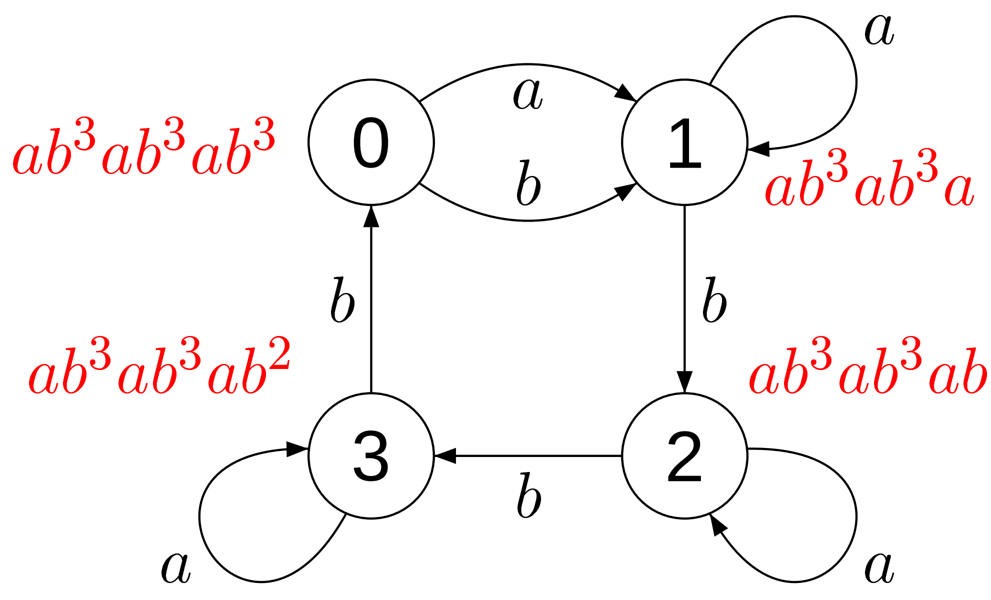
Strongly connected digraphs

Thus, we assume that our synchronizing automata are strongly connected as digraphs.

Observe that such an automaton can be reset to any state. That is, to every state q of the automaton one can assign an instruction (a reset word) w_q such that following w_q one will surely arrive at q from any initial state.

Strongly connected digraphs

Thus, we assume that our synchronizing automata are strongly connected as digraphs.
Observe that such an automaton can be reset to any state. That is, to every state q of the automaton one can assign an instruction (a reset word) w_q such that following w_q one will surely arrive at q from any initial state.

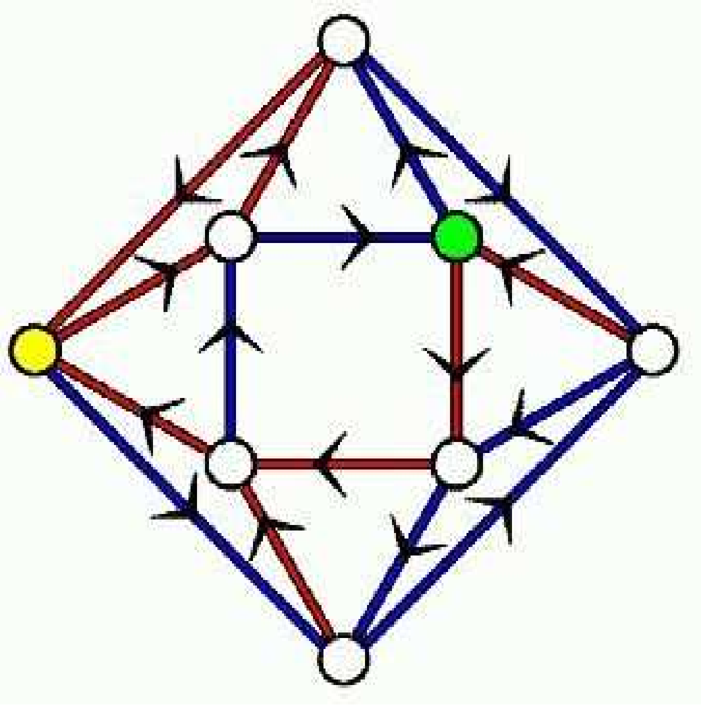


Road Coloring

Now think of the automaton as of a scheme of a transport network in which arrows correspond to roads and labels are treated as colors of the roads.

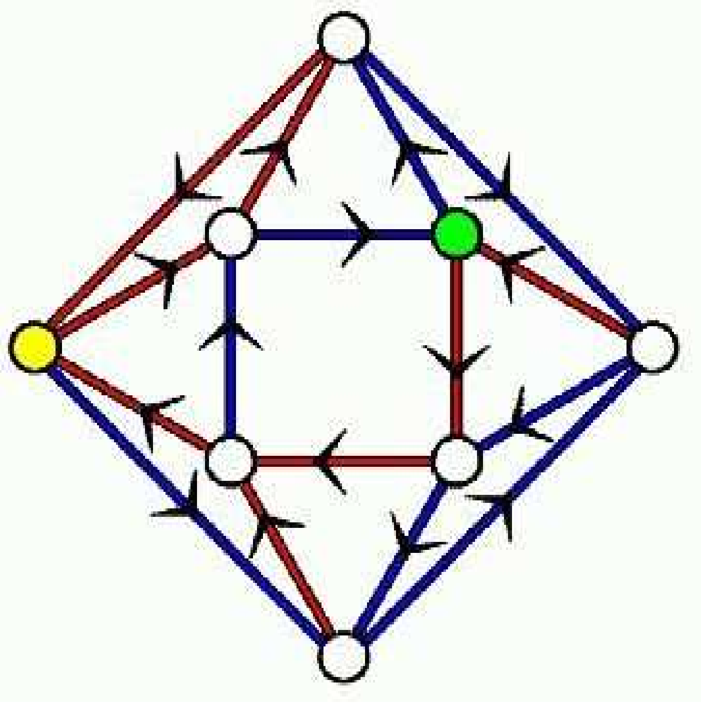
Road Coloring

Now think of the automaton as of a scheme of a transport network in which arrows correspond to roads and labels are treated as colors of the roads.



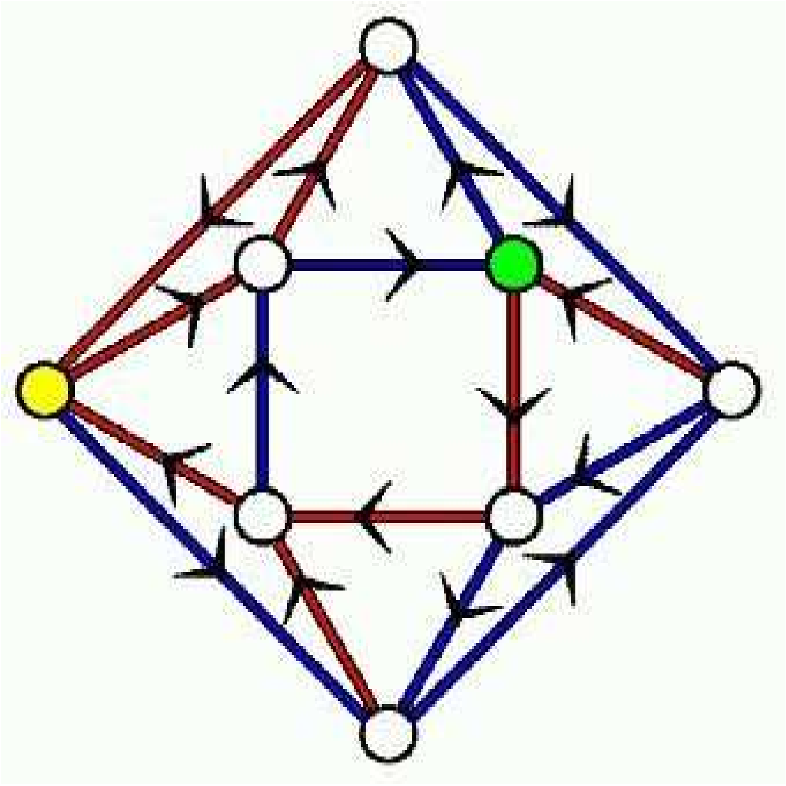
Road Coloring

Now think of the automaton as of a scheme of a transport network in which arrows correspond to roads and labels are treated as colors of the roads.

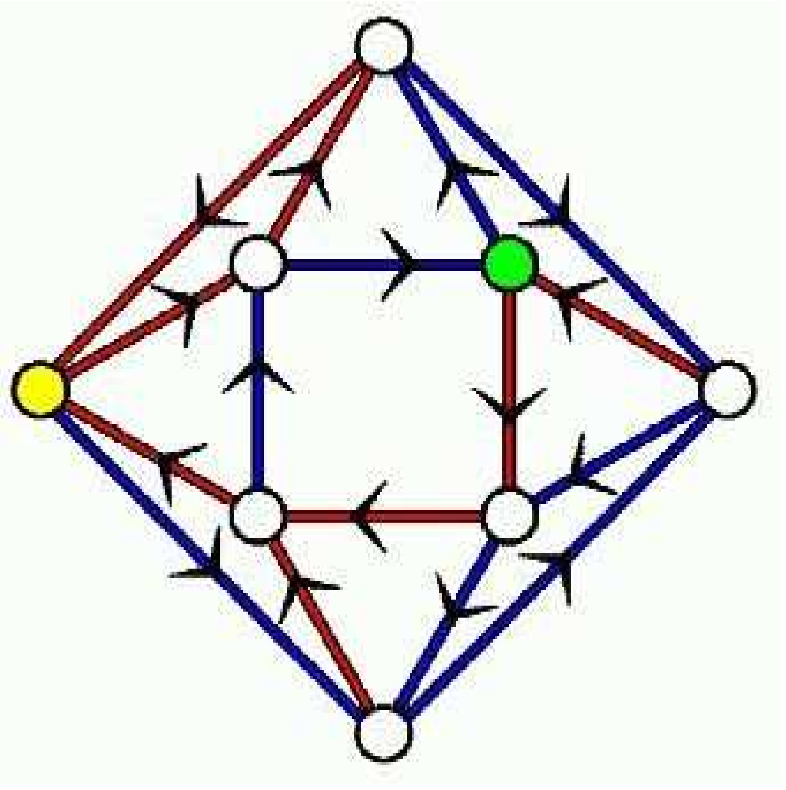


Then for each node there is a sequence of colors that brings one to the chosen node from anywhere.

Road Coloring

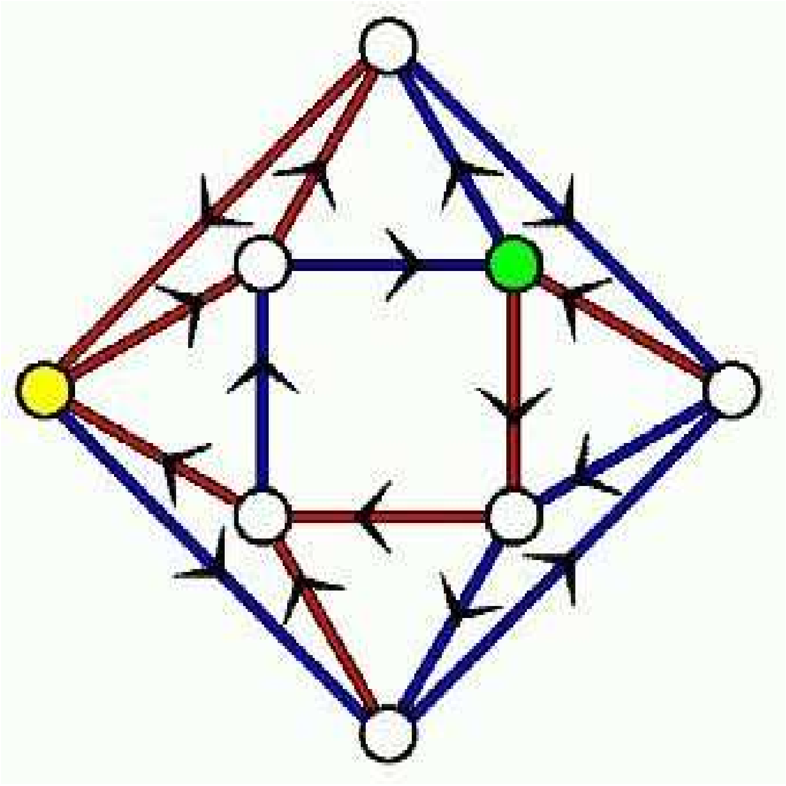


Road Coloring



For the green node:
blue-blue-red-blue-blue-red-blue-blue-red.

Road Coloring



For the green node:
blue-blue-red-blue-blue-red-blue-blue-red.

For the yellow node:
blue-red-red-blue-red-red-blue-red-red.

Road Coloring

Now suppose that we have a transport network, that is, a strongly connected digraph.

Road Coloring

Now suppose that we have a transport network, that is, a strongly connected digraph.

We aim to help people to orientate in it, and as we have seen, a neat solution may consist in coloring the roads such that our digraph becomes a synchronizing automaton.

Road Coloring

Now suppose that we have a transport network, that is, a strongly connected digraph.

We aim to help people to orientate in it, and as we have seen, a neat solution may consist in coloring the roads such that our digraph becomes a synchronizing automaton. When is such a coloring possible?

Road Coloring

Now suppose that we have a transport network, that is, a strongly connected digraph.

We aim to help people to orientate in it, and as we have seen, a neat solution may consist in coloring the roads such that our digraph becomes a synchronizing automaton. When is such a coloring possible?
In other words: which strongly connected digraphs may appear as underlying digraphs of synchronizing automata?

Road Coloring

Now suppose that we have a transport network, that is, a strongly connected digraph.

We aim to help people to orientate in it, and as we have seen, a neat solution may consist in coloring the roads such that our digraph becomes a synchronizing automaton. When is such a coloring possible?
In other words: which strongly connected digraphs may appear as underlying digraphs of synchronizing automata?

An obvious necessary condition:
all vertices should have the same out-degree.
In what follows we refer to this as to the constant out-degree condition.

Primitivity

A less obvious necessary condition is called aperiodicity or primitivity:
the g.c.d. of lengths of all cycles should be equal to 1.

Primitivity

A less obvious necessary condition is called aperiodicity or primitivity:
the g.c.d. of lengths of all cycles should be equal to 1.

To see why primitivity is necessary, suppose that $\Gamma = (V, E)$ is a strongly connected digraph and $k > 1$ is a common divisor of lengths of its cycles.

Primitivity

A less obvious necessary condition is called aperiodicity or primitivity:
the g.c.d. of lengths of all cycles should be equal to 1.

To see why primitivity is necessary, suppose that $\Gamma = (V, E)$ is a strongly connected digraph and $k > 1$ is a common divisor of lengths of its cycles. Take a vertex $v_0 \in V$ and, for $i = 0, 1, \ldots, k-1$, let

$$V_i = \{v \in V \mid \exists \text{ path from } v_0 \text{ to } v \text{ of length } i \pmod{k}\}.$$

Primitivity

A less obvious necessary condition is called aperiodicity or primitivity:
the g.c.d. of lengths of all cycles should be equal to 1.

To see why primitivity is necessary, suppose that $\Gamma = (V, E)$ is a strongly connected digraph and $k > 1$ is a common divisor of lengths of its cycles. Take a vertex $v_0 \in V$ and, for $i = 0, 1, \ldots, k-1$, let

$$V_i = \{v \in V \mid \exists \text{ path from } v_0 \text{ to } v \text{ of length } i \pmod{k}\}.$$

Clearly, $V = \bigcup_{i=0}^{k-1} V_i$.

Primitivity

A less obvious necessary condition is called aperiodicity or primitivity:
the g.c.d. of lengths of all cycles should be equal to 1.

To see why primitivity is necessary, suppose that $\Gamma = (V, E)$ is a strongly connected digraph and $k > 1$ is a common divisor of lengths of its cycles. Take a vertex $v_0 \in V$ and, for $i = 0, 1, \ldots, k-1$, let

$$V_i = \{v \in V \mid \exists \text{ path from } v_0 \text{ to } v \text{ of length } i \pmod k\}.$$

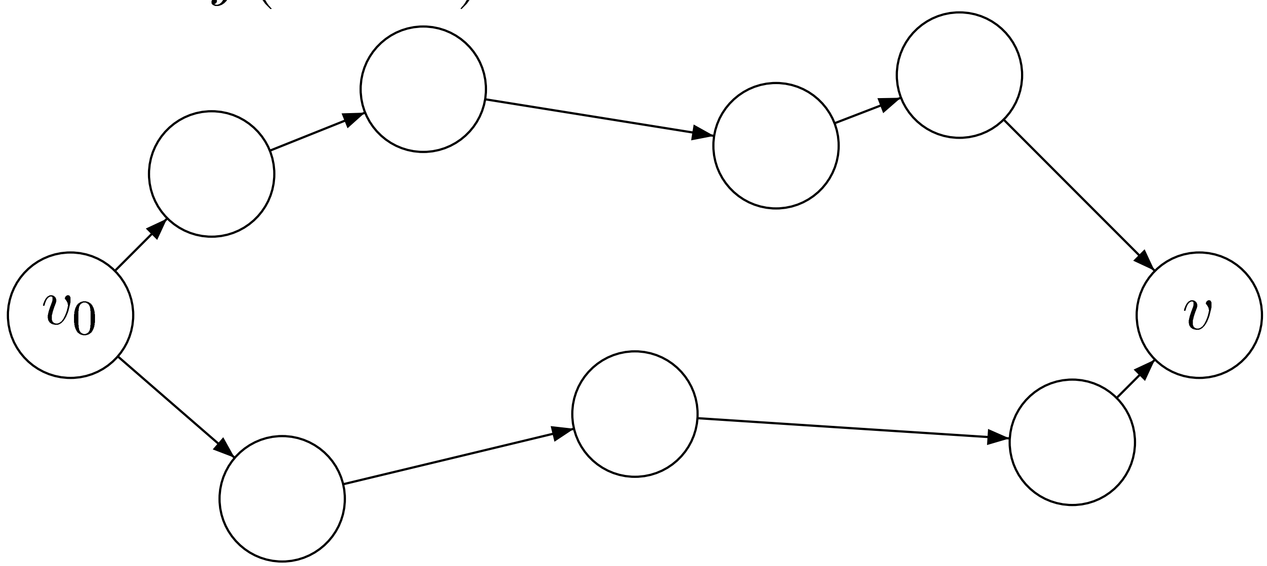
Clearly, $V = \bigcup_{i=0}^{k-1} V_i$. We claim that $V_i \cap V_j = \varnothing$ if $i \neq j$.

Primitivity

Let $v \in V_i \cap V_j$ where $i \neq j$. This means that in Γ there are two paths from v_0 to v: of length $\ell \equiv i \pmod{k}$ and of length $m \equiv j \pmod{k}$.

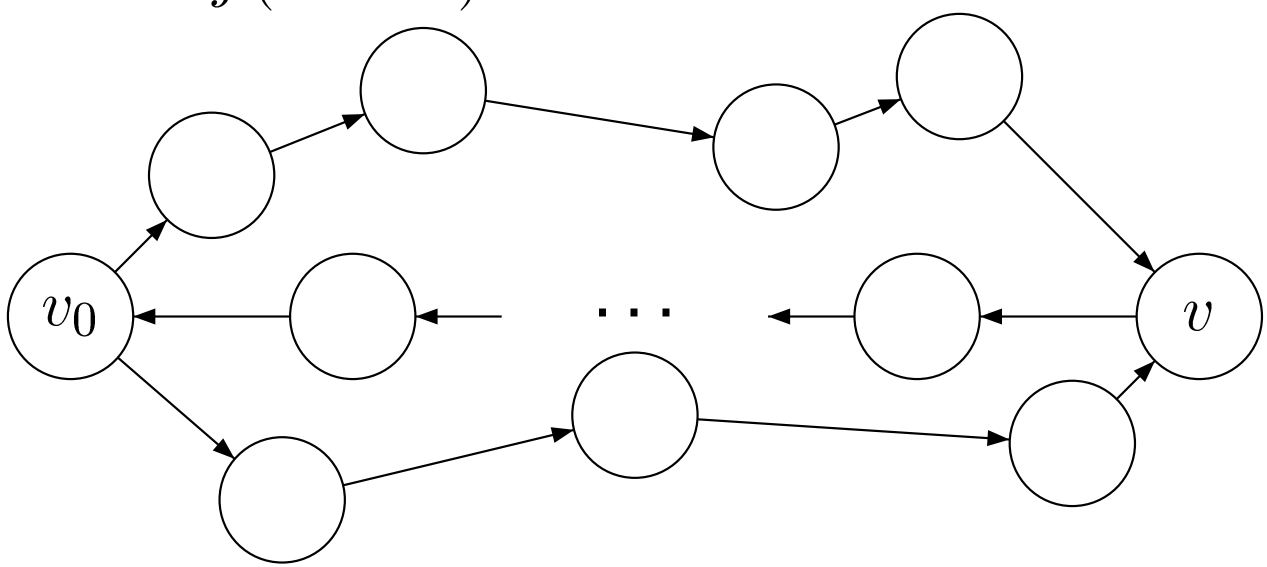
Primitivity

Let $v \in V_i \cap V_j$ where $i \neq j$. This means that in Γ there are two paths from v_0 to v: of length $\ell \equiv i \pmod{k}$ and of length $m \equiv j \pmod{k}$.



Primitivity

Let $v \in V_i \cap V_j$ where $i \neq j$. This means that in Γ there are two paths from v_0 to v: of length $\ell \equiv i \,(\mathrm{mod}\ k)$ and of length $m \equiv j \,(\mathrm{mod}\ k)$.



There is also a path v to v_0 of length, say, n. Combining it with the two paths above we get a cycle of length $\ell + n$ and a cycle of length $m + n$.

Primitivity

Since k divides the length of any cycle in Γ, we have $\ell + n \equiv i + n \equiv 0 \pmod{k}$ and $m + n \equiv j + n \equiv 0 \pmod{k}$, whence $i \equiv j \pmod{k}$, a contradiction.

Primitivity

Since k divides the length of any cycle in Γ, we have $\ell + n \equiv i + n \equiv 0 \pmod{k}$ and $m + n \equiv j + n \equiv 0 \pmod{k}$, whence $i \equiv j \pmod{k}$, a contradiction.

Thus, V is a disjoint union of $V_0, V_1, \ldots, V_{k-1}$, and by the definition each arrow in Γ leads from V_i to $V_{i+1 \pmod{k}}$.

Primitivity

Since k divides the length of any cycle in Γ, we have $\ell + n \equiv i + n \equiv 0 \pmod{k}$ and $m + n \equiv j + n \equiv 0 \pmod{k}$, whence $i \equiv j \pmod{k}$, a contradiction.

Thus, V is a disjoint union of $V_0, V_1, \ldots, V_{k-1}$, and by the definition each arrow in Γ leads from V_i to $V_{i+1 \pmod{k}}$.

Then Γ definitely cannot be converted into a synchronizing automaton by any labelling of its arrows: for instance, no paths of the same length ℓ originated in V_0 and V_1 can terminate in the same vertex because they end in $V_{\ell \pmod{k}}$ and in $V_{\ell+1 \pmod{k}}$ respectively.

Road Coloring Conjecture

The Road Coloring Conjecture claims that the two necessary conditions (constant out-degree and primitivity) are in fact sufficient.

Road Coloring Conjecture

The Road Coloring Conjecture claims that the two necessary conditions (constant out-degree and primitivity) are in fact sufficient. In other words: *every strongly connected primitive digraph with constant out-degree admits a synchronizing coloring*.

Road Coloring Conjecture

The Road Coloring Conjecture claims that the two necessary conditions (constant out-degree and primitivity) are in fact sufficient. In other words: *every strongly connected primitive digraph with constant out-degree admits a synchronizing coloring*.

The Road Coloring Conjecture was explicitly stated by Adler, Goodwyn and Weiss in 1977 (Equivalence of topological Markov shifts, Israel J. Math., 27, 49–63).

Road Coloring Conjecture

The Road Coloring Conjecture claims that the two necessary conditions (constant out-degree and primitivity) are in fact sufficient. In other words: *every strongly connected primitive digraph with constant out-degree admits a synchronizing coloring*.

The Road Coloring Conjecture was explicitly stated by Adler, Goodwyn and Weiss in 1977 (Equivalence of topological Markov shifts, Israel J. Math., 27, 49–63). In an implicit form it was present already in an earlier memoir by Adler and Weiss (Similarity of automorphisms of the torus, Memoirs Amer. Math. Soc., 98 (1970)) almost 40 years ago.

Road Coloring Conjecture

The original motivation for the Road Coloring Conjecture comes from symbolic dynamics, see Marie-Pierre Béal and Dominiues Perrin's chapter "Symbolic Dynamics and Finite Automata" in Handbook of Formal Languages, Vol.I.

Road Coloring Conjecture

The original motivation for the Road Coloring Conjecture comes from symbolic dynamics, see Marie-Pierre Béal and Dominiues Perrin’s chapter “Symbolic Dynamics and Finite Automata” in Handbook of Formal Languages, Vol.I.

However the conjecture is natural also from the viewpoint of the “reverse engineering” of synchronizing automata as presented here.

Road Coloring Conjecture

The original motivation for the Road Coloring Conjecture comes from symbolic dynamics, see Marie-Pierre Béal and Dominiues Perrin's chapter "Symbolic Dynamics and Finite Automata" in Handbook of Formal Languages, Vol.I.

However the conjecture is natural also from the viewpoint of the "reverse engineering" of synchronizing automata as presented here.

The Road Coloring Conjecture has attracted much attention. There were several interesting partial results, and finally the problem was solved (in the affirmative) in August 2007 by Avraham Trahtman.

Road Coloring Conjecture

The original motivation for the Road Coloring Conjecture comes from symbolic dynamics, see Marie-Pierre Béal and Dominiues Perrin's chapter "Symbolic Dynamics and Finite Automata" in Handbook of Formal Languages, Vol.I.

However the conjecture is natural also from the viewpoint of the "reverse engineering" of synchronizing automata as presented here.

The Road Coloring Conjecture has attracted much attention. There were several interesting partial results, and finally the problem was solved (in the affirmative) in August 2007 by Avraham Trahtman. Trahtman's solution got much publicity this year.

Stability

Trahtman's proof heavily depends on a neat idea of stability which is due to Karel Culik II, Juhani Karhumäki and Jarkko Kari (A note on synchronized automata and Road Coloring Problem, Int. J. Found. Comput. Sci., 13 (2002) 459–471).

Stability

Trahtman's proof heavily depends on a neat idea of stability which is due to Karel Culik II, Juhani Karhumäki and Jarkko Kari (A note on synchronized automata and Road Coloring Problem, Int. J. Found. Comput. Sci., 13 (2002) 459–471). Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA. We define the relation $\sim$ on Q as follows:

$$q \sim q' \iff \forall u \in \Sigma^* \; \exists v \in \Sigma^* \; q\,.\,uv = q'.uv.$$

Stability

Trahtman's proof heavily depends on a neat idea of stability which is due to Karel Culik II, Juhani Karhumäki and Jarkko Kari (A note on synchronized automata and Road Coloring Problem, Int. J. Found. Comput. Sci., 13 (2002) 459–471). Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA. We define the relation $\sim$ on Q as follows:

$$q \sim q' \iff \forall u \in \Sigma^* \ \exists v \in \Sigma^* \ q\,.\,uv = q'.uv.$$

$\sim$ is called the *stability relation* and any pair (q, q') such that $q \sim q'$ is called *stable*.

Stability

Trahtman's proof heavily depends on a neat idea of stability which is due to Karel Culik II, Juhani Karhumäki and Jarkko Kari (A note on synchronized automata and Road Coloring Problem, Int. J. Found. Comput. Sci., 13 (2002) 459–471). Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA. We define the relation $\sim$ on Q as follows:

$$q \sim q' \iff \forall u \in \Sigma^* \ \exists v \in \Sigma^* \ q\,.\,uv = q'.uv.$$

$\sim$ is called the *stability relation* and any pair (q, q') such that $q \sim q'$ is called *stable*. It is immediate that $\sim$ is a congruence of the automaton $\mathscr{A}$.

Stability

Trahtman's proof heavily depends on a neat idea of stability which is due to Karel Culik II, Juhani Karhumäki and Jarkko Kari (A note on synchronized automata and Road Coloring Problem, Int. J. Found. Comput. Sci., 13 (2002) 459–471). Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA. We define the relation $\sim$ on Q as follows:

$$q \sim q' \iff \forall u \in \Sigma^* \ \exists v \in \Sigma^* \ q \, . \, uv = q'.uv.$$

$\sim$ is called the *stability relation* and any pair (q, q') such that $q \sim q'$ is called *stable*. It is immediate that $\sim$ is a congruence of the automaton $\mathscr{A}$. Also observe that $\mathscr{A}$ is synchronizing iff all pairs are stable.

Stability

We say that a coloring of a digraph with constant out-degree is *stable* if the resulting automaton contains at least one stable pair (q, q') with $q \neq q'$.

Stability

We say that a coloring of a digraph with constant out-degree is *stable* if the resulting automaton contains at least one stable pair (q, q') with $q \neq q'$.The crucial observation by Culik, Karhumäki and Kari is

Proposition CKK. *Suppose every strongly connected primitive digraph with constant out-degree and more than 1 vertex has a stable coloring. Then the Road Coloring Conjecture holds true.*

Stability

We say that a coloring of a digraph with constant out-degree is *stable* if the resulting automaton contains at least one stable pair (q, q') with $q \neq q'$.The crucial observation by Culik, Karhumäki and Kari is

Proposition CKK. *Suppose every strongly connected primitive digraph with constant out-degree and more than 1 vertex has a stable coloring. Then the Road Coloring Conjecture holds true.*

The proof is rather straightforward: one inducts on the number of vertices in the digraph.

Stability

We say that a coloring of a digraph with constant out-degree is *stable* if the resulting automaton contains at least one stable pair (q, q') with $q \neq q'$.The crucial observation by Culik, Karhumäki and Kari is

Proposition CKK. *Suppose every strongly connected primitive digraph with constant out-degree and more than 1 vertex has a stable coloring. Then the Road Coloring Conjecture holds true.*

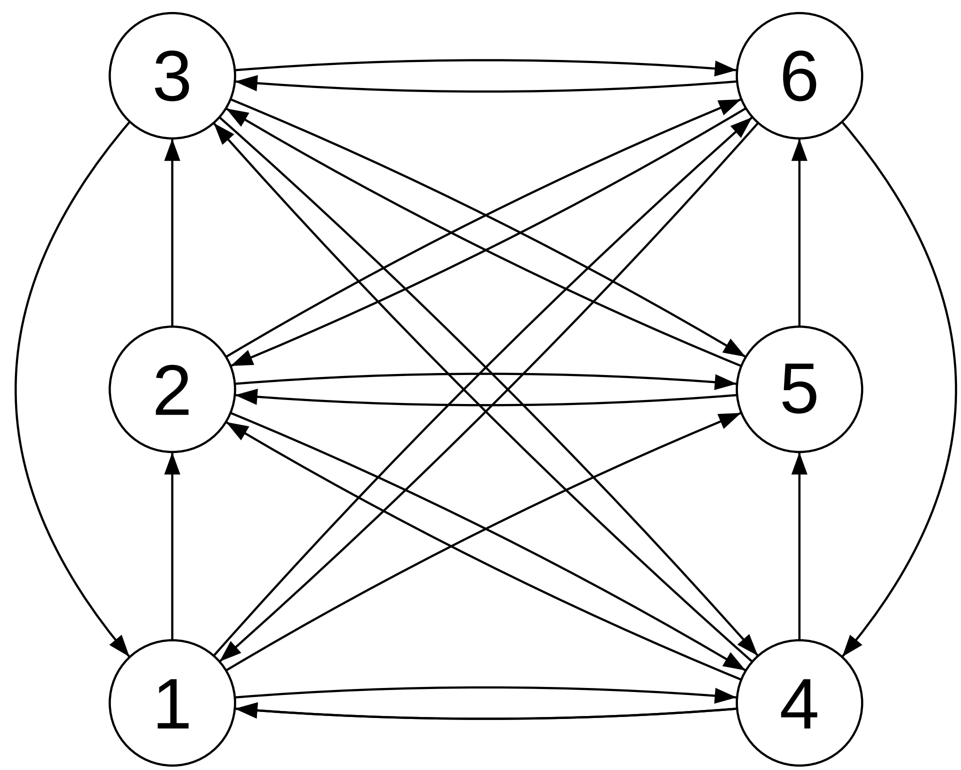
The proof is rather straightforward: one inducts on the number of vertices in the digraph. If Γ admits a stable coloring and $\mathscr{A}$ is the resulting automaton, then the quotient automaton $\mathscr{A}/\sim$ admits a synchronizing recoloring by the induction assumption.

Stability

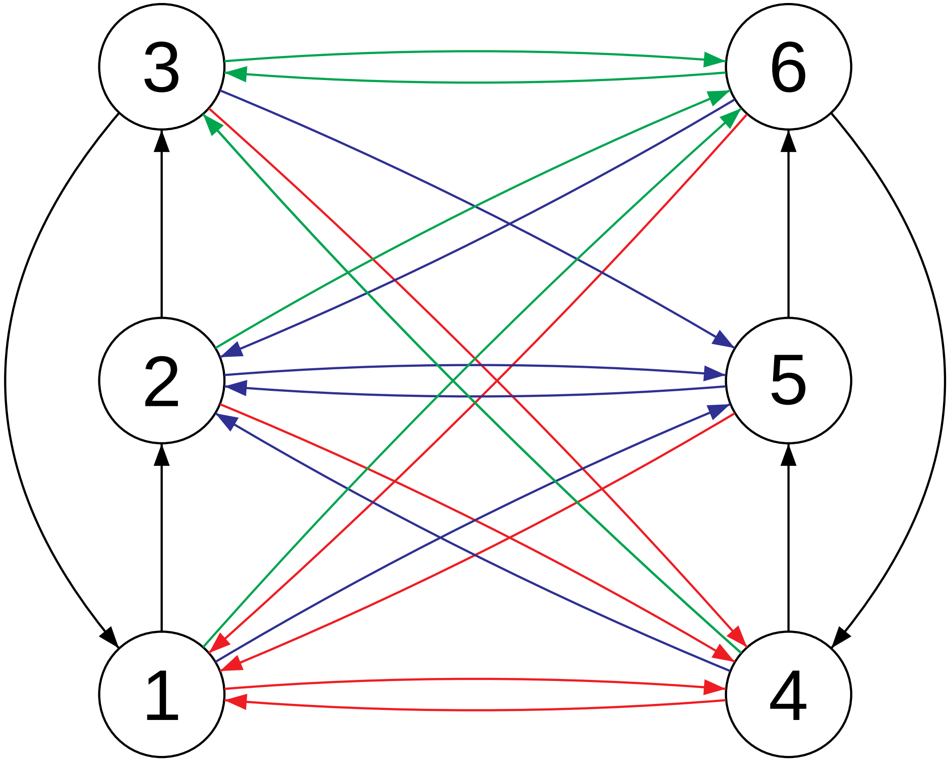
Then it remains to lift the correct coloring of $\mathscr{A} / \sim$ to a synchronizing coloring of Γ.

Stability

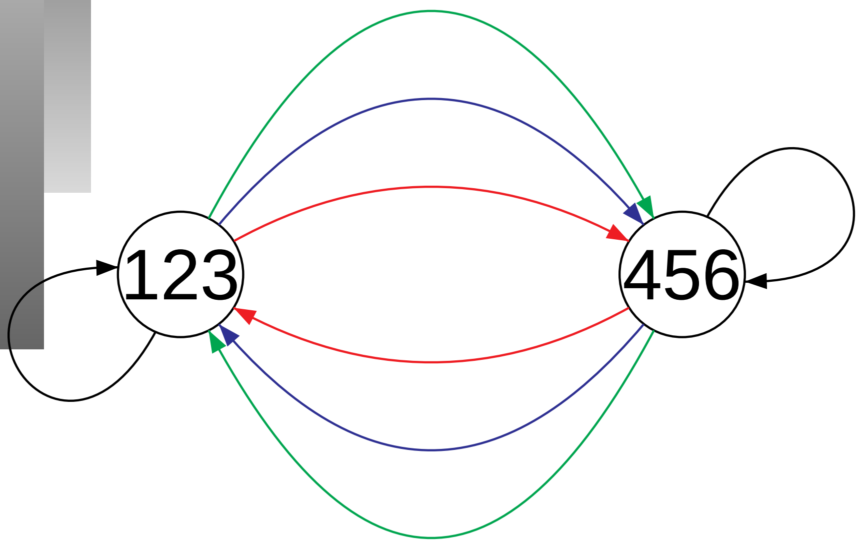
Then it remains to lift the correct coloring of $\mathscr{A}/\sim$ to a synchronizing coloring of Γ.



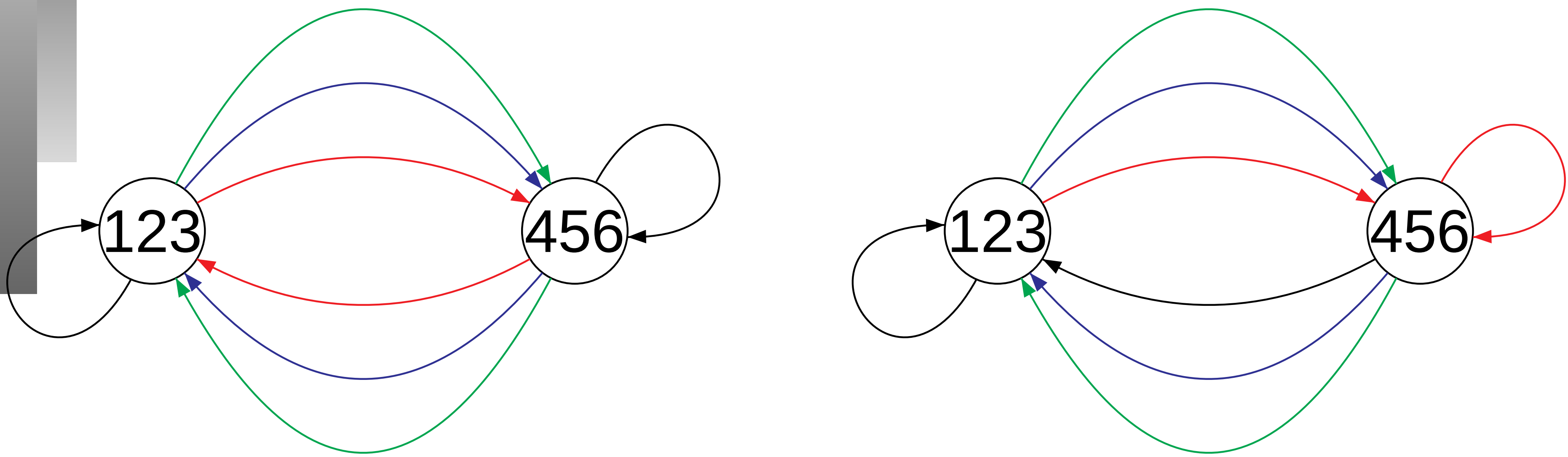
Then it remains to lift the correct coloring of $\mathscr{A}/\sim$ to a synchronizing coloring of Γ.



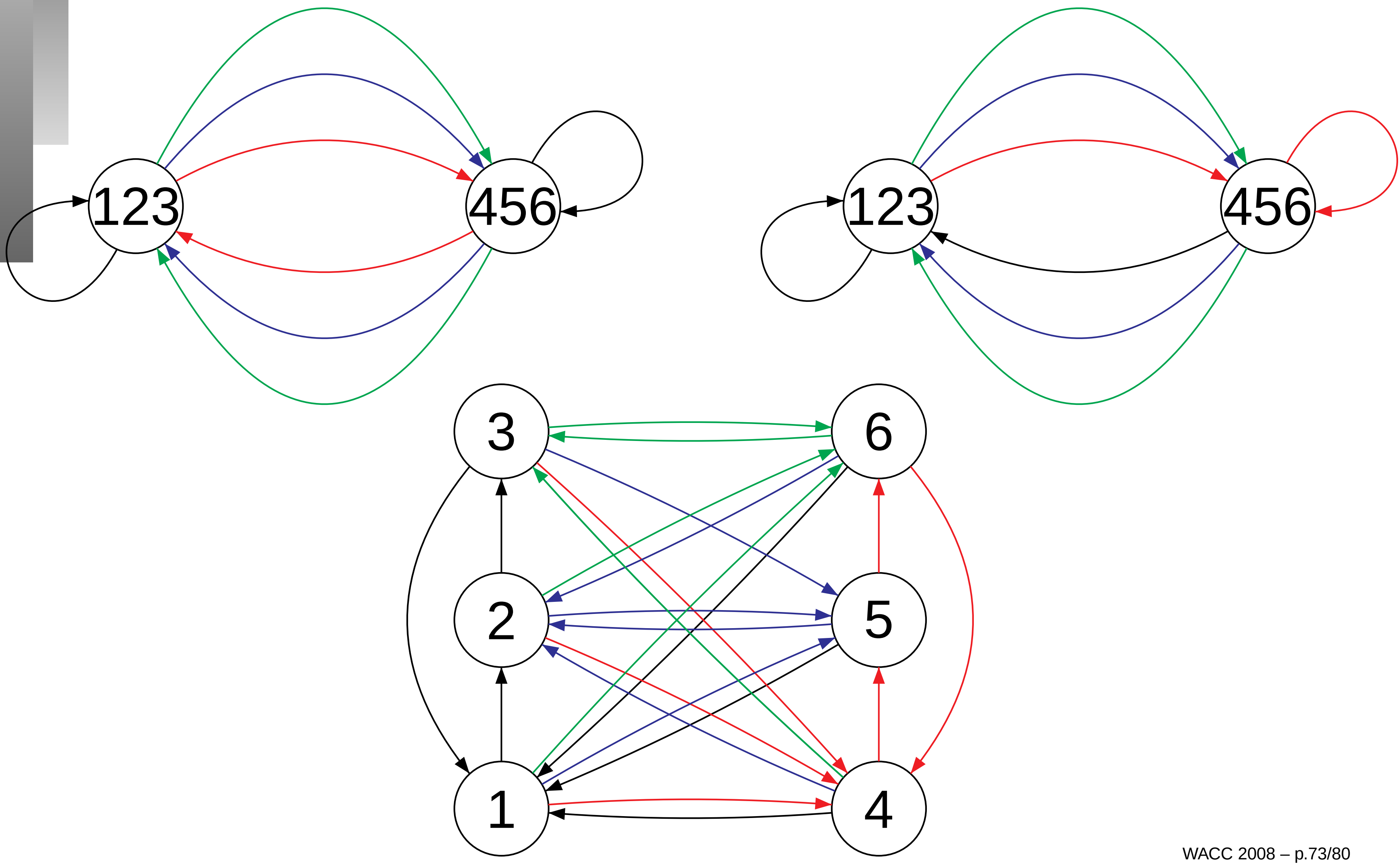
Stability



Stability



Stability



Road Coloring Conjecture

Trahtman has managed to prove exactly what was needed to use Proposition CKK: every strongly connected primitive digraph with constant out-degree and more than 1 vertex has a stable coloring. Thus, Road Coloring Conjecture holds true.

Road Coloring Conjecture

Trahtman has managed to prove exactly what was needed to use Proposition CKK: every strongly connected primitive digraph with constant out-degree and more than 1 vertex has a stable coloring. Thus, Road Coloring Conjecture holds true.

The proof is not difficult but still a bit too technical for a presentation at the end of our conference.

Summary of Open Problems

- The real potential of the greedy algorithm.

Summary of Open Problems

- The real potential of the greedy algorithm. Is it true that, for any synchronizing automaton with n states, the greedy algorithm returns a reset word of length $\Omega((n^2 \log n)$? Is the same is true for a polynomial algorithm, even non-deterministic?

Summary of Open Problems

- The real potential of the greedy algorithm. Is it true that, for any synchronizing automaton with n states, the greedy algorithm returns a reset word of length $\Omega((n^2 \log n)$? Is the same is true for a polynomial algorithm, even non-deterministic?

- Synchronizing random automata: what is the expectation of the minimum length of reset words for the random automaton with n states?

Summary of Open Problems

- The real potential of the greedy algorithm. Is it true that, for any synchronizing automaton with n states, the greedy algorithm returns a reset word of length $\Omega((n^2 \log n)$? Is the same is true for a polynomial algorithm, even non-deterministic?
- Synchronizing random automata: what is the expectation of the minimum length of reset words for the random automaton with n states? What is the probability distribution of this random variable?

Summary of Open Problems

- The Černý Conjecture: is it true that, for any synchronizing automaton with n states, there exists a reset word of length $(n-1)^2$?

Summary of Open Problems

- The Černý Conjecture: is it true that, for any synchronizing automaton with n states, there exists a reset word of length $(n-1)^2$?
- The Rank Conjecture: is it true that, for any DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ with $\boldsymbol{n}$ states and rank $\boldsymbol{k}$, there is a $\boldsymbol{w} \in \boldsymbol{\Sigma}^*$ of length $(\boldsymbol{n} - \boldsymbol{k})^{\boldsymbol{2}}$ such that $|\boldsymbol{Q} \,.\, \boldsymbol{w}| = \boldsymbol{k}$?

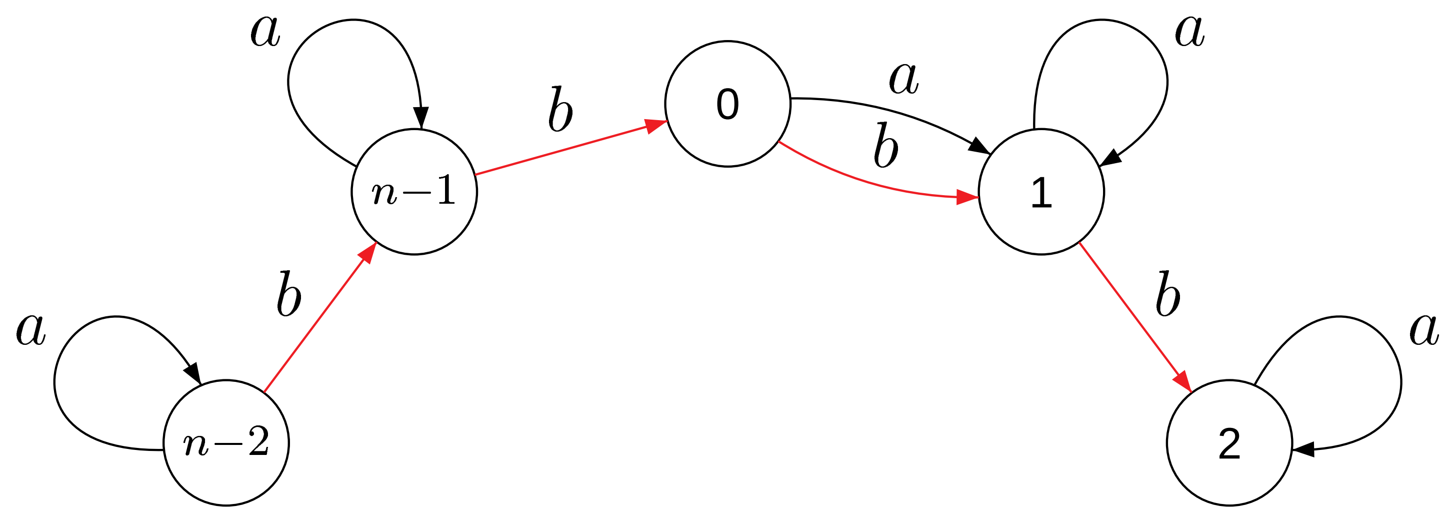
Summary of Open Problems

- The Černý Conjecture: is it true that, for any synchronizing automaton with n states, there exists a reset word of length $(n-1)^2$?
- The Rank Conjecture: is it true that, for any DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with n states and rank k, there is a $w \in \Sigma^*$ of length $(n-k)^2$ such that $|Q \, . \, w| = k$?
- The hybrid Černý/Road Coloring problem. Let Γ be a strongly connected primitive digraph with constant out-degree and n vertices. What is the minimum length of reset words for synchronizing colorings of Γ?

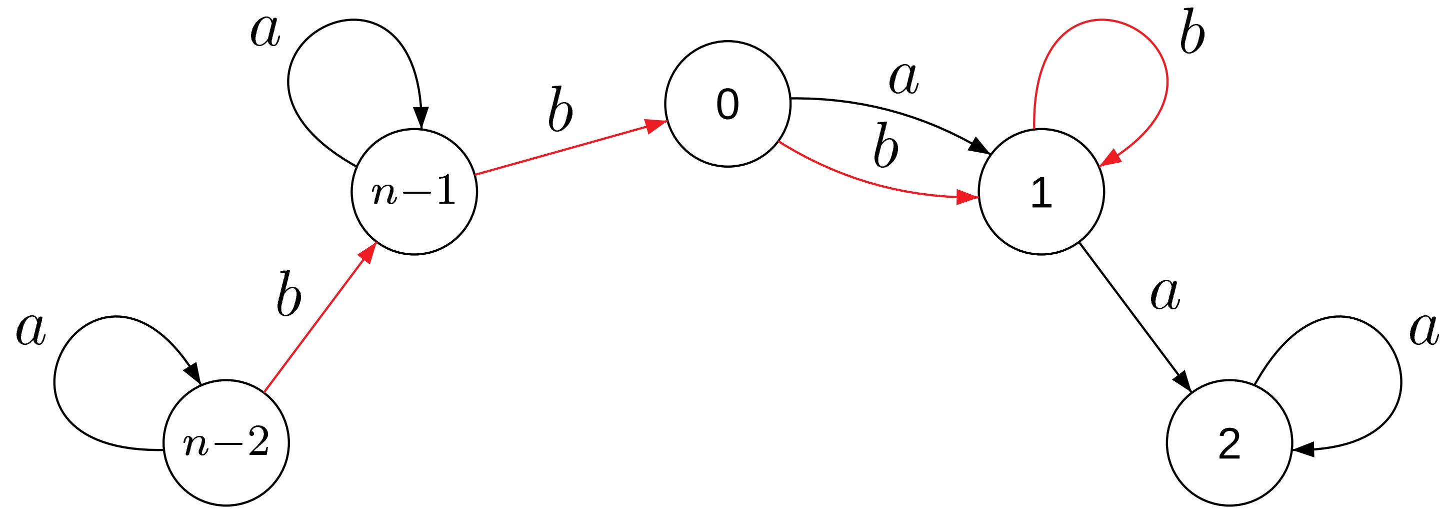
Summary of Open Problems

- The Černý Conjecture: is it true that, for any synchronizing automaton with n states, there exists a reset word of length $(n-1)^2$?
- The Rank Conjecture: is it true that, for any DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with n states and rank k, there is a $w \in \Sigma^*$ of length $(n-k)^2$ such that $|Q\,.\,w| = k$?
- The hybrid Černý/Road Coloring problem. Let Γ be a strongly connected primitive digraph with constant out-degree and n vertices. What is the minimum length of reset words for synchronizing colorings of Γ? For instance, the Černý automata admit synchronizing recolorings with pretty short reset words.

Summary of Open Problems

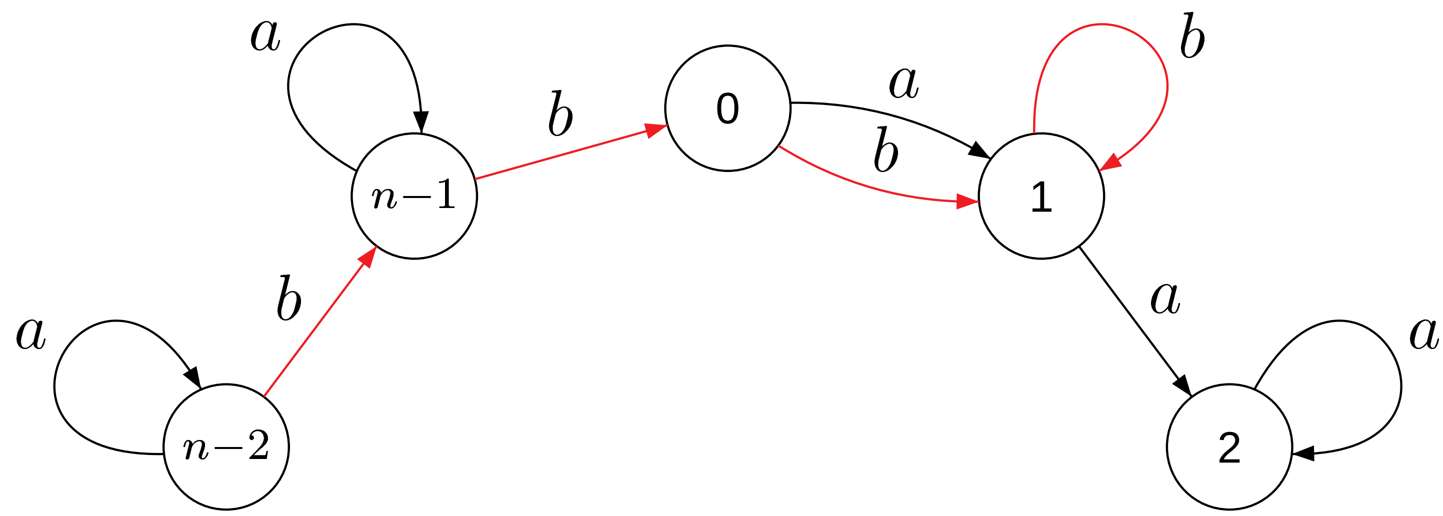


Summary of Open Problems



The recolored automaton is reset by the word b^{n-1}.

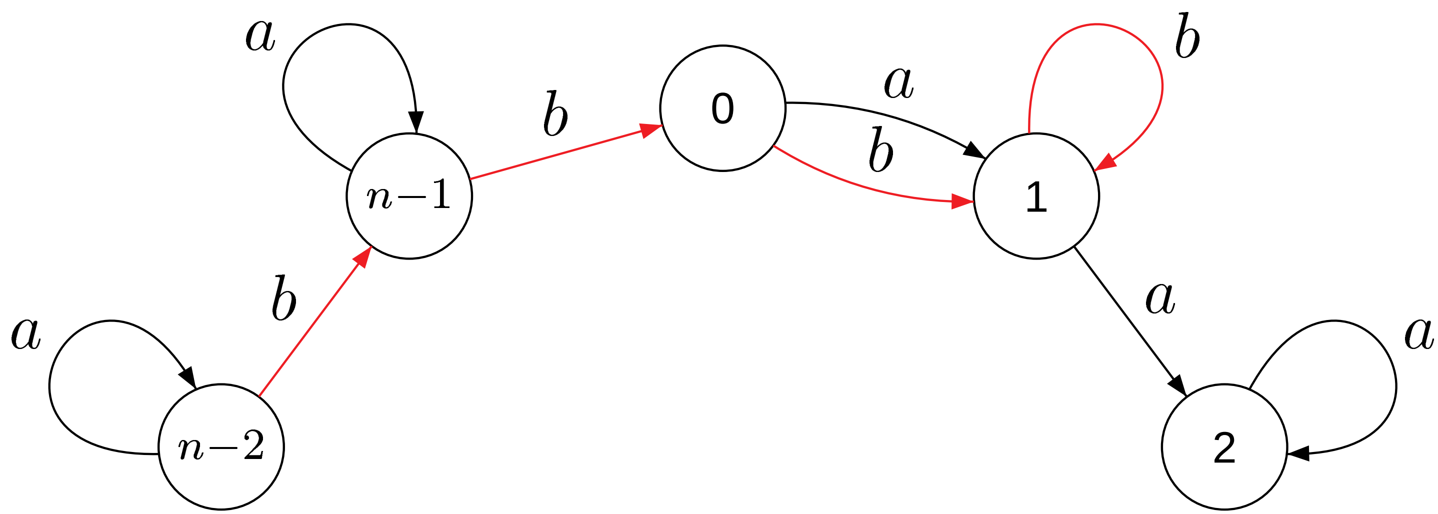
Summary of Open Problems



The recolored automaton is reset by the word b^{n-1}.

• Careful Road Coloring Problem. From the viewpoint of transportation network the constant out-degree condition does not seem to be natural.

Summary of Open Problems



The recolored automaton is reset by the word b^{n-1}.

• Careful Road Coloring Problem. From the viewpoint of transportation network the constant out-degree condition does not seem to be natural. We rather want to find a synchronizing coloring for arbitrary strongly connected primitive digraph Γ, the number of colors being the maximal out-degree of Γ.

Summary of Open Problems

But in the absence of the constant out-degree condition, the resulting automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is incomplete. We need a suitable modification of the notion of a synchronizing automaton for this case.

Summary of Open Problems

But in the absence of the constant out-degree condition, the resulting automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is incomplete. We need a suitable modification of the notion of a synchronizing automaton for this case.

We say that $w = a_1 \cdots a_\ell$ with $a_1, \ldots, a_\ell \in \Sigma$ is a *careful reset word* for $\mathscr{A}$ if
- $\delta(q, a_1)$ is defined for all $q \in Q$,
- $\delta(q, a_i)$ with $1 < i \leq \ell$ is defined for all

$$q \in Q \,.\, a_1 \cdots a_{i-1},$$

- $|Q \,.\, w| = 1$.

Summary of Open Problems

But in the absence of the constant out-degree condition, the resulting automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is incomplete. We need a suitable modification of the notion of a synchronizing automaton for this case.

We say that $w = a_1 \cdots a_\ell$ with $a_1, \ldots, a_\ell \in \Sigma$ is a *careful reset word* for $\mathscr{A}$ if
- $\delta(q, a_1)$ is defined for all $q \in Q$,
- $\delta(q, a_i)$ with $1 < i \leq \ell$ is defined for all $q \in Q \, . \, a_1 \cdots a_{i-1}$,
- $|Q \, . \, w| = 1$.

In transport network terms this means that following the instruction w is always possible and brings one to the node which is independent of the initial node.

Summary of Open Problems

The automaton $\mathscr{A}$ is then said to be *carefully synchronizing*.

Summary of Open Problems

The automaton $\mathscr{A}$ is then said to be *carefully synchronizing*.

The Careful Road Coloring Problem asks under which conditions strongly connected digraphs admit carefully synchronizing colorings.

Summary of Open Problems

The automaton $\mathscr{A}$ is then said to be *carefully synchronizing*.

The Careful Road Coloring Problem asks under which conditions strongly connected digraphs admit carefully synchronizing colorings. Is it true that every primitive strongly connected digraph has such a coloring? (The Careful Road Coloring Conjecture)

Thanks

Thank you very much for your kind attention!

Thanks

Thank you very much for your kind attention!

Thank you very much for your kind attention!

Thanks

Thank you very much for your kind attention!

Thank you very much for your kind attention!

Thank you very much for your kind attention!